

# 易语言

YI

YU

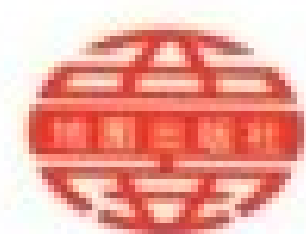
YAN

# 编程系统

BIAN CHENG XI TONG

全中文全可视跨平台编程

易语言教材编委会 编著



西安地图出版社



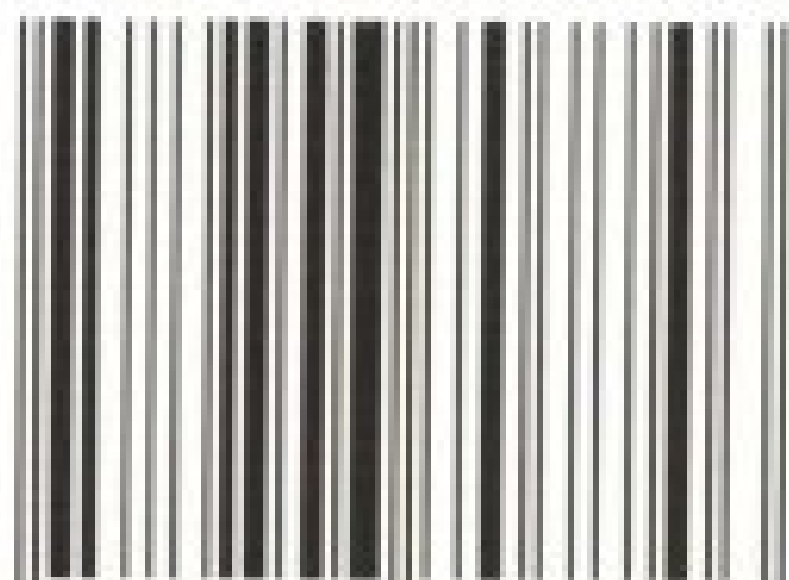
责任编辑：史 英  
封面设计：张志恒

## 内 容 简 介

YIYU YAN

本书是一本全面介绍全中文全可视跨平台编程工具——“易语言”专著。“易语言”实现了真正的汉语编程，彻底摆脱了使用传统英文编程工具所存在的语言障碍和思维模式障碍，以更加符合中国人习惯的方式编写真正的“中国制造”程序。开发编程无需再记忆繁杂的英文命令与单词，只要会输入汉字或拼音就能开发出专业化的程序。“易语言”内置桌面型数据库，支持多种大型数据库，可实现网络及硬件通信编程。面向对象的编程理念、丰富的界面元素、数千条系统命令、API与COM支持，足以满足各个行业不同应用层次的开发要求。“易语言”将是广大电脑用户的理想编程工具。全书由30章和3个附录组成。主要内容包括：易语言概述、数据类型、变量、常量、资源、命令、流程控制命令、子程序、窗口组件、多媒体、网络与通讯、系统控制、易数据库、外部数据库、API调用、易模块、DLL的编写与调用、OCX组件与类型库、COM对象、面向对象编程、Linux程序编写、数据结构支持库、数据操作支持库、文本语音转换支持库、电话语音支持库、数码设备支持库、脚本语言支持库、办公支持库，以及如何进行程序调试、编译与发布、易向导使用。本书内容通俗易懂，资料丰富翔实，图文并茂，编程技巧简单实用。适合初、中级编程用户，同时也可作为高等院校各专业教学、自学参考书以及社会培训班指导用书。读者在使用本书过程中的技术问题，请在易语言技术交流论坛（[www.dywt.com.cn](http://www.dywt.com.cn)）上提出。

ISBN 7-80670-774-3



9 787806 707746 >

ISBN 7-80670-774-3/TP·27

定价：65.00元

# 易语言

YI

YU

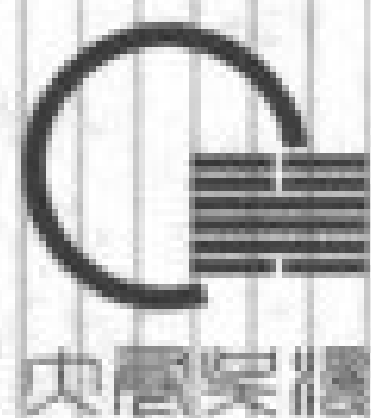
YAN

# 编程系统

BIAN CHENG XI TONG

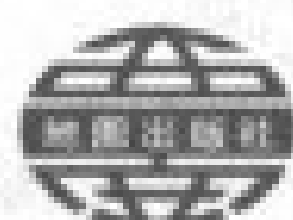
全中文全可视跨平台编程

易语言教材编委会 编著



大连大有吴涛易语言软件开发有限公司

DALIAN DAYOU WUTAO YIYUYAN SOFTWARE DEVELOPMENT CO., LTD



西安地图出版社

图书在版编目 (CIP) 数据

易语言编程系统/易语言教材编委会编著. —西安: 西安地图出版社, 2005. 3  
ISBN 7-80670-774-3

I. 易… II. 易… III. 汉语—程序语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2005) 第 019258 号

易语言编程系统

易语言教材编委会 编著

西安地图出版社出版发行

(西安市友谊东路 334 号 邮政编码: 710054)

新华书店经销 河南第一新华印刷厂 印刷

787 毫米×1092 毫米 1/16 开本 印张: 34 字数: 864 千字

2005 年 3 月第 1 版 2005 年 3 月第 1 次印刷

印数: 0001-5000 册

ISBN 7-80670-774-3/TP·27

定价: 65.00 元



# 易

祝易语言的创造者们互拜  
拓计算机通及使用的天地  
中建功立业

張政祥



1998年十月



易難為



# 前 言

易语言汉语编程环境是吴涛先生于 2000 年开始自主开发，历经 4 年后（2004 年 3 月）由大连大有房地产开发公司投入资金，与吴涛共同组建了大连大有吴涛易语言软件开发有限公司。在吴涛已有开发工作的基础上，公司以实现易语言汉语编程环境的产品化和产业化为目标，增加了资金投入，扩大了技术开发队伍，着力推进易语言汉语编程环境的产品化和产业化以及市场的推广应用。

易语言汉语编程环境的试用版是以共享软件的方式提供给社会，公司通过自己的网站 ([www.dywt.com.cn](http://www.dywt.com.cn)) 向社会开放。国内对易语言汉语编程表现出浓厚的兴趣，具有比较广泛的群众基础，自 2000 年以来通过网上访问、下载和其他方式接触、了解、学习、使用易语言编程的客户群保守估计超过 40 万人，可下载的网站已达上千家，包括台湾、新加坡。

在易语言论坛上正式注册的用户超过 20000 人，每天同时在线人数达 300 人以上，最高峰时超过 1000 人。其中不少人学习易语言后，用易语言编出了易语言程序。目前已经组织了一个“易语言俱乐部论坛”，并举办了两次“易语言大奖赛”。经过统计，由用户上传到“易语言俱乐部论坛”上并已经公开的易语言源代码有 5 万个以上。

然而，在市场上还缺少有关的培训教材和使用手册，2003 年 1 月曾出版了一本《易语言即学即用教程》，但由于版本的更新，此书已不合适了，为此公司内部组织成立易语言图书编撰小组着手编写新的学习教材，命名为《易语言编程系统》。

易语言是一种汉语编程语言，易语言汉语编程环境的目标为：自主设计并实现一种实用且易学易用的汉语计算机编程语言及其编程环境，并实现产品化。形成一个在编程语言领域里具有完全自主知识产权的集成化开发平台。我国在六十年代计算所三室曾经设计并实现了一种类 Fortran 的汉语编程算法语言，在三室中沿用了十多年历史，随着 Fortran、Basic 及 C++ 语言的引入，中国再也没有一个自主开发的编程语言被推广而被大家接受，而易语言在四年前从网上发布，很快被大家接，多少国内的一些著名专家，一旦他了解了易语言之后，无不赞不绝口，为我国能有自主版权又适合我国国情的编程语言而高兴。

本书以深入浅出地全面介绍了“易语言及其编程环境的功能”，图文并茂，且提供了大量实例，几乎包含了当前所有的编程言的功能和编程技术，而且还有不少创新之处。

今受公司诚邀，为本书进行审校的工作，进一步加深了对易语言的了解，图书的作者做了大量的工作。由此我们对此评价是：易语言汉语编程环境的推广应用将为我国的计算机普及提供一个新通道，为非计算机编程人员掌握编程技术提供一座新桥梁，为初级、高级及不同领域的编程人员提供了完美的开发和维护平台。

易语言公司图书编委会成员有：高庆狮（中科院院士）、仲萃豪（中科院软件所研究员）、曹东启（中科院软件所研究员）、吴涛、关天人、邢长斌、王军，在此向所有合作出书人员表示感谢。但还可能有不尽之处，敬请指正。

仲萃豪 于 北京

# 目 录

## 前言

第一章 易语言概述.....	1
1.1 易语言概要介绍.....	1
1.1.1 易语言简介.....	1
1.1.2 易语言的特点.....	2
1.2 易语言的安装.....	4
1.3 易语言的基本操作界面.....	6
1.3.1 易语言的界面.....	6
1.3.2 如何在设计窗口中添加组件.....	8
1.3.3 如何使用易语言帮助系统.....	8
1.3.4 如何配置易语言.....	9
1.4 易语言代码输入技巧.....	10
1.4.1 内置输入法.....	10
1.4.2 系统输入法.....	11
1.4.3 参数分步输入.....	12
1.4.4 输入备注与代码屏蔽.....	12
1.4.5 四种输入语句分类.....	13
1.4.6 易语言中的关键字.....	13
1.5 第一个易程序.....	14
1.5.1 易程序的结构.....	14
1.5.2 开始写第一个易程序.....	14
1.5.3 分析第一个易程序.....	16
1.6 易程序进阶.....	16
1.7 本章小结.....	18
第二章 数据类型、运算符与表达式.....	19
2.1 易语言的数据类型.....	19
2.2 基本数据类型.....	19
2.2.1 了解基本数据类型.....	19
2.2.2 给变量和返回值定义数据类型.....	20
2.2.3 给数据正确赋值.....	21
2.2.4 数据的比较.....	22
2.2.5 数据类型的存储字节与溢出.....	23
2.3 特殊数据类型.....	25
2.3.1 了解特殊数据类型.....	25
2.3.2 动态添加组件.....	25

2.4 自定义数据类型.....	25
2.5 易语言运算符.....	27
2.6 运算符与表达式.....	28
2.6.1 算术运算符和算术表达式 .....	28
2.6.2 赋值运算符和赋值表达式 .....	29
2.7 本章小结.....	29
第三章 变量、常量与资源.....	31
3.1 变量.....	31
3.1.1 了解变量 .....	31
3.1.2 变量类型 .....	32
3.1.3 变量的赋值 .....	34
3.1.4 变量的初始值 .....	34
3.2 静态局部变量.....	35
3.3 数组变量.....	36
3.3.1 数组变量的定义及赋值 .....	37
3.3.2 动态管理数组变量 .....	39
3.4 易语言常量.....	43
3.4.1 了解常量 .....	43
3.4.2 ASCII 码 .....	43
3.4.3 常量的使用 .....	44
3.4.4 枚举常量及使用方法 .....	45
3.4.5 自定义常量及使用方法 .....	47
3.5 易语言资源表.....	48
3.5.1 向资源表中添加资源 .....	48
3.5.2 使用资源表中的资源 .....	50
3.5.3 将资源表中的资源导出 .....	51
3.5.4 向资源表中导入可执行文件 .....	51
3.6 本章小结.....	52
第四章 常用命令.....	53
4.1 了解易语言命令.....	53
4.1.1 命令的格式 .....	53
4.1.2 即时帮助和帮助文档 .....	53
4.1.3 命令的返回值 .....	55
4.1.4 命令的套用 .....	56
4.1.5 数组类型的参数与返回值 .....	57
4.2 流程控制命令.....	58
4.2.1 了解流程控制类命令 .....	58
4.2.2 分支类流程控制命令 .....	59
4.2.3 循环类流程控制命令 .....	61
4.2.4 跳转类流程控制命令 .....	65



4.3 其他常用命令 .....	67
4.3.1 文本操作类命令 .....	67
4.3.2 时间操作类命令 .....	68
4.3.3 位运算命令 .....	69
4.3.4 其他常用命令 .....	72
4.4 本章小结 .....	74
第五章 子程序的编写与调用 .....	75
5.1 子程序的初步应用 .....	75
5.1.1 子程序的分类 .....	75
5.1.2 用户自定义子程序的创建 .....	75
5.1.3 子程序的调用 .....	77
5.1.4 返回值和参数的定义 .....	77
5.1.5 编写一个子程序 .....	77
5.1.6 子程序指针 .....	79
5.2 子程序的参数属性 .....	80
5.2.1 参数的“参考”属性 .....	80
5.2.2 参数的“可空”属性 .....	81
5.2.3 参数的“数组”属性 .....	83
5.3 编写寻找文件子程序 .....	84
5.4 本章小结 .....	87
第六章 窗口组件、菜单和对话框 .....	88
6.1 窗口组件 .....	88
6.1.1 窗口的基本属性 .....	88
6.1.2 窗口的基本事件 .....	91
6.1.3 增加新窗口和弹出窗口 .....	94
6.1.4 窗口的重要方法 .....	96
6.2 编辑菜单 .....	100
6.2.1 新建菜单 .....	100
6.2.2 菜单的热键及属性 .....	101
6.2.3 弹出菜单 .....	103
6.3 对话框 .....	105
6.3.1 “信息框 ( )”命令 .....	105
6.3.2 “信息框 ( )”命令的返回值 .....	106
6.3.3 “输入框 ( )”命令 .....	107
6.3.4 “输入框 ( )”命令的应用 .....	107
6.4 本章小结 .....	108
第七章 组件介绍 .....	110
7.1 易语言组件简介 .....	110
7.1.1 易语言内部组件 .....	110
7.2 组件的属性 .....	113

7.2.1 组件的共有属性 .....	113
7.2.2 组件的专有属性 .....	114
7.2.3 动态修改组件属性 .....	121
7.3 组件的专有方法 .....	123
7.4 事件的触发 .....	125
7.4.1 事件子程序 .....	125
7.4.2 组件事件的应用 .....	126
7.5 本章小结 .....	131
第八章 多媒体 .....	132
8.1 声音（音频） .....	133
8.1.1 媒体播放命令 .....	133
8.2 图片处理（图形图像） .....	135
8.2.1 图片的合并 .....	135
8.2.2 图片框组件 .....	136
8.3 动画 .....	140
8.3.1 图片框的 GIF 动画 .....	140
8.3.2 窗口动画 .....	140
8.4 图片转场 .....	142
8.5 影视 .....	145
8.5.1 影像框组件 .....	145
8.5.2 高级影像框组件 .....	145
8.5.3 外部影像组件 .....	146
8.6 本章小结 .....	148
第九章 网络与通讯 .....	149
9.1 基础知识 .....	149
9.2 网络应用型程序 .....	150
9.2.1 网络组件 .....	150
9.2.2 互联网支持库 .....	159
9.3 数据通讯程序 .....	171
9.3.1 网络通信命令 .....	172
9.3.2 数据通讯组件 .....	172
9.3.3 网络通讯支持库 .....	179
9.4 硬件通信型程序 .....	183
9.4.1 硬件通信组件 .....	184
9.4.2 端口访问支持库 .....	187
9.5 本章小结 .....	188
第十章 系统控制 .....	189
10.1 运行命令 .....	189
10.2 系统信息类命令 .....	190
10.3 配置文件和注册表 .....	191

10.3.1 配置文件.....	191
10.3.2 调用系统配置工具.....	193
10.3.3 注册表.....	194
10.4 本章练习.....	203
第十一章 易数据库.....	204
11.1 易数据库文件的组成.....	204
11.2 使用工具创建和维护数据库.....	205
11.3 用命令创建数据库.....	207
11.3.1 编程前的准备.....	208
11.3.2 创建数据库.....	209
11.3.3 打开数据库.....	209
11.3.4 置当前数据库.....	210
11.3.5 关闭数据库.....	211
11.4 用程序维护数据库.....	211
11.4.1 记录操作.....	211
11.4.2 当前记录指针.....	212
11.4.3 读写字段.....	213
11.4.4 添加记录.....	213
11.4.5 删除记录.....	213
11.4.6 查找记录.....	214
11.5 数据库实例.....	215
11.6 数据库相关组件.....	218
11.6.1 通用提供者、数据库提供者.....	219
11.6.2 数据源.....	220
11.6.3 表格.....	227
11.6.4 数据库相关组件的应用.....	230
11.6.5 制作表格模板.....	232
11.7 本章小结.....	234
第十二章 外部数据库调用.....	235
12.1 外部数据库相关知识.....	235
12.1.1 易外部数据库组件简介.....	235
12.1.2 ODBC 与 ADO 简介.....	235
12.1.3 SQL 语言简介.....	237
12.1.4 常用的 SQL 语句.....	237
12.2 ODBC 连接数据库组件.....	243
12.2.1 “外部数据库” 组件属性.....	243
12.2.2 “外部数据库” 组件重要方法.....	243
12.2.3 “外部数据库提供者” 组件.....	245
12.3 ADO 操作数据库组件.....	245
12.3.1 “数据库连接” 组件.....	246

12.3.2 “记录集” 组件 .....	248
12.4 外部数据库应用 .....	252
12.4.1 外部数据库操作例程 .....	252
12.4.2 用表格组件显示数据库 .....	258
12.5 Access 数据库 .....	259
12.5.1 Access 数据库简介 .....	259
12.5.2 Access 数据库综合例程 .....	259
12.6 SQL Server 数据库 .....	266
12.6.1 SQL Server 简介 .....	266
12.6.2 SQL Server 2000 安装 .....	268
12.6.3 创建 SQL Server 2000 数据库及表 .....	270
12.6.4 使用易语言操作 SQL SERVER 数据库 .....	273
12.7 MYSQL 数据库 .....	278
12.7.1 MYSQL 常用命令 .....	279
12.7.2 易语言操作 MYSQL 数据库 .....	282
12.8 本章小结 .....	291
第十三章 API 的调用 .....	292
13.1 API 简介 .....	292
13.2 API 的定义 .....	292
13.3 API 的应用 .....	294
13.3.1 内部 API .....	295
13.3.2 外部 API .....	301
13.4 本章小结 .....	304
第十四章 易模块 .....	305
14.1 易模块的作用 .....	305
14.2 易模块的调用方法 .....	305
14.3 易模块的开发与编译 .....	309
14.3.1 易模块的开发 .....	309
14.3.2 易模块的编译 .....	311
14.3.3 易模块的改写实例 .....	313
14.4 本章小结 .....	315
第十五章 DLL 的编写与调用 .....	317
15.1 DLL 与 API 函数的关系 .....	317
15.2 DLL 的开发与编译 .....	317
15.3 调用 DLL 的方法 .....	320
15.4 DLL 应用实例 .....	321
15.5 本章小结 .....	324
第十六章 OCX 组件与类型库 .....	325
16.1 OCX 组件 .....	325
16.1.1 OCX 的安装 .....	325



16.1.2 OCX 的汉化 .....	328
16.1.3 OCX 的使用方法 .....	330
16.2 类型库的封装和使用 .....	342
16.2.1 类型库的封装 .....	343
16.2.2 类型库的使用 .....	350
16.3 本章小结 .....	353
第十七章 COM 对象 .....	354
17.1 基本概念 .....	354
17.1.1 什么是 COM .....	354
17.1.2 COM 对象 .....	354
17.1.3 COM 接口 .....	354
17.2 COM 对象的使用 .....	355
17.2.1 新的数据类型“对象” .....	355
17.2.2 使用 COM 对象的一般步骤 .....	357
17.2.3 一个完整的例子 .....	358
17.3 本章小结 .....	360
第十八章 易语言面向对象编程 .....	361
18.1 基础知识 .....	361
18.1.1 类的概念 .....	361
18.1.2 类和对象的关系 .....	362
18.1.3 类中私有成员的特性 .....	364
18.1.4 派生类和继承性 .....	365
18.1.5 子类中直接调用父类方法 .....	366
18.1.6 类的多态性与虚拟方法 .....	367
18.2 类的实际应用例程 .....	371
18.3 本章小结 .....	378
第十九章 Linux 程序编写 .....	379
19.1 Linux 简介 .....	379
19.2 创建 Linux 程序 .....	379
19.3 Linux 程序开发与运行 .....	380
19.3.1 了解例程的相关情况 .....	381
19.3.2 例程服务端代码讲解 .....	381
19.3.3 例程客户端代码讲解 .....	385
19.3.4 编译与运行 .....	387
19.4 本章小结 .....	388
第二十章 数据结构支持库 .....	389
20.1 数据结构基础 .....	389
20.1.1 节点 .....	389
20.1.2 链表 .....	389
20.1.3 栈 .....	391

20.1.4 队列.....	392
20.1.5 树.....	393
20.1.6 二叉树.....	394
20.2 栈的应用例程.....	395
第二十一章 数据操作支持库.....	400
21.1 加密技术.....	400
21.1.1 数据加密.....	400
21.1.2 数据认证.....	401
21.2 数据校验.....	402
21.2.1 “数字签名()”命令.....	403
21.2.2 “签名验证()”命令.....	404
21.2.3 “取数据摘要()”命令.....	405
第二十二章 数值计算支持库.....	407
22.1 数值计算支持库简介.....	407
22.2 数值计算支持库的各数据类型.....	407
22.2.1 复数运算.....	407
22.2.2 矩阵运算.....	408
22.2.3 傅立叶变换.....	409
22.2.4 微积分.....	410
22.2.5 概要统计.....	410
22.2.6 联立方程.....	411
22.2.7 曲线拟和.....	412
22.2.8 大数.....	412
22.2.9 其他计算.....	413
22.2.10 算式解析.....	413
22.3 大数计算器.....	413
第二十三章 文本语音转换支持库.....	417
23.1 文本语音转换简介.....	417
23.2 机读文本.....	417
23.3 语音识别.....	421
23.4 本章小结.....	425
第二十四章 电话语音支持库.....	426
24.1 支持库简介.....	426
24.2 支持库重要方法.....	426
24.3 支持库相关例程.....	429
第二十五章 数码设备支持库.....	436
25.1 支持库简介.....	436
25.2 支持库属性与方法.....	436
25.2.1 数码设备的重要属性.....	436
25.2.2 数码设备的方法.....	437

25.3 支持库相关例程.....	438
25.4 视频设备.....	442
第二十六章 脚本语言支持组件.....	445
26.1 组件简介.....	445
26.2 属性和方法.....	445
26.3 组件应用实例.....	446
26.3.1 四则表达式计算器.....	446
26.3.2 外部程序调用.....	447
第二十七章 Word 2000 支持库.....	449
27.1 Word 2000 支持库简介.....	449
27.2 Word 2000 支持库组件.....	449
27.2.1 “Word 程序” 组件.....	449
27.2.2 “Word 文档集” 组件.....	451
27.2.3 “Word 图形” 组件.....	455
27.3 Word 2000 支持库例程.....	456
第二十八章 Excel 2000 支持库.....	462
28.1 Excel 2000 支持库简介.....	462
28.2 Excel 2000 支持库的组件.....	462
28.2.1 “Excel 程序” 组件.....	462
28.2.2 “Excel 工作簿” 组件.....	463
28.2.3 “Excel 图表” 组件.....	463
28.3 Excel 2000 支持库例程.....	464
第二十九章 PowerPoint 2000 支持库.....	471
29.1 PowerPoint 2000 支持库简介.....	471
29.2 PowerPoint 2000 支持库的组件.....	471
29.2.1 “PPT 程序” 组件.....	471
29.2.2 “PPT 文稿” 组件.....	472
29.2.3 “PPT 播放” 组件.....	475
29.3 PowerPoint 2000 支持库例程.....	476
29.3.1 人工切换.....	476
29.3.2 自动播放.....	477
第三十章 办公组件支持库.....	480
30.1 办公组件简介.....	480
30.2 办公组件属性.....	480
30.3 办公组件的方法.....	481
30.3.1 办公组件的重要方法.....	481
30.3.2 办公组件中接口对象的重要方法.....	484
30.4 办公组件事件.....	488
30.5 办公组件例程.....	488
30.6 本章小结.....	497

附录一 程序调试.....	498
调试工具.....	498
调试输出命令.....	503
调试应用总结.....	506
附录二 易语言编译与发布.....	507
非独立编译.....	507
独立编译.....	508
程序发布.....	509
更改图标样式.....	512
附录三 易语言向导.....	514
易向导的作用及意义.....	514
易向导的使用方法.....	514
易向导的编写.....	517
1. “写出程序（）”命令 .....	518
2. “删除程序（）”和“删除程序段（）”命令 .....	519
3. “修改程序（）”命令 .....	520
4. “置组件属性（）”命令 .....	520
5. “清除修改记录（）”命令 .....	520
6. “定义模板变量（）”命令 .....	520



# 第一章 易语言概述

## 1.1 易语言概要介绍

### 1.1.1 易语言简介

易语言是一种汉语编程语言，由大连大有吴涛易语言软件开发有限公司出品。易语言采用全中文汉语编程，该开发环境是建立在 Windows 平台上，支持全中文、可视化编程操作，功能丰富且易学易用，可以满足国内各类计算机用户的需求。并可直接在 Windows 环境下开发 Linux 程序。

作为一款全中文的编程语言，易语言融入了中华文化和民俗习惯，用户不再需要按照国外的语言习惯、表达方式、甚至是思维方式而是直接用中文，按照我们自己的习惯去编写程序。易语言编程环境方便直观、快捷实用，不但支持程序代码可以全部用中文来编写，并且操作界面亦为全中文。即使一个根本不懂英文或者对英文了解很少的初级用户也能够快速进入计算机程序编写的大门。甚至初中或小学文化水平的人也可以较快地学会编制一些简单程序。

易语言并不是把现有的编程工具简单地进行表面汉化或封装而成的，它拥有自己独立的高质量编译器，中文源代码被直接编译为目的机器的 CPU 指令。值得称道的是，其编译器所编译出来的可执行代码与操作系统平台无关，因此能够很方便地实现跨平台编程。目前，易语言可同时支持 Windows 和 Linux 程序的开发，今后移植到其他操作系统平台也非常方便，使之不再依赖特定的操作系统环境，这也符合国家发展开发自主知识产权基础系统软件的战略部署，易语言编程环境本身就是一个重要的基础系统软件，而且还为其他自主知识产权操作系统提供了配套、合适的应用软件开发工具。

易语言的可视化设计操作相对其他编程语言来说优势十分明显，它不仅仅支持程序窗体界面设计的可视化操作，连代码流程图都完整的嵌入到程序代码设计操作之中，这是一般任何编程软件所不具备的功能。同时，它内置了一种专用、记忆式中文输入法，支持中文语句快速录入，即时命令函数提示功能，彻底突破了中文语句输入速度的瓶颈。

易语言支持模块化开发，可满足大型应用软件系统协同开发的要求。它也可以编译出符合标准 Win32 DLL 的程序模块，供其他的易程序，甚至是 VC、Delphi、VB 程序在自身代码中直接调用。易语言内置的“易模块”功能，也是易语言模块化开发中的一个重要组成部分，易程序可以直接在程序中引用编译好的易模块，进一步简化了易程序的开发复杂程度。

易语言自带的易数据库，能够充分满足开发桌面型数据库程序的需要。同时，易语言





提供对 ADO、ODBC 等数据库接口技术的全面支持，并可直接访问 MySQL 数据库，因此能够很好的与各种外部数据库进行数据交换，便于开发和实现基于大型数据库的应用软件系统。

易语言能够充分利用现有的一切编程资源，提供对 OCX 控件、类型库、API 函数、COM 协议等一系列接口的全面支持，并开放其支持库接口技术文档供第三方使用，利用这些不计其数的资源，极大地提升了易语言的实用功能。

易语言支持当今先进的编程理念，譬如面向对象的程序编写方法、面向事件的消息处理机制等等，易语言与其他编程语言是一种融合互通的关系，了解了易语言对了解其他编程语言具有极大的帮助。易语言支持用户定义和使用对象，支持类的构造、析构、继承、虚拟方法、多态、封装等特性。

易语言新版本推出了更多的扩展支持库，如数据操作支持库、数值计算支持库、文本语音转换支持库、电话语音支持库、数码设备支持库、脚本语言支持组件、办公类支持库等，极大地方便了用户编写程序，新的行业支持库还在不断地开发中。

易语言除简体中文版外，还提供繁体中文版、日文版和英文版等多种语言版本，非常适合多民族本土化开发的要求。本书中如无特别说明，全部是以简体中文版作为讲解对象。

## 1.1.2 易语言的特点

### 1. 全可视化

一般的可视化编程语言，仅支持图形用户界面的可视化设计操作，而易语言除了支持此类可视化操作，还支持程序流程的即时可视化呈视，极大地减少了程序录入错误。即：用户在编写程序的过程中，可以即时看到当前程序的运行流程及路线，有助于培养编程思路，提高解决编程问题的能力。如图 1-1 所示。

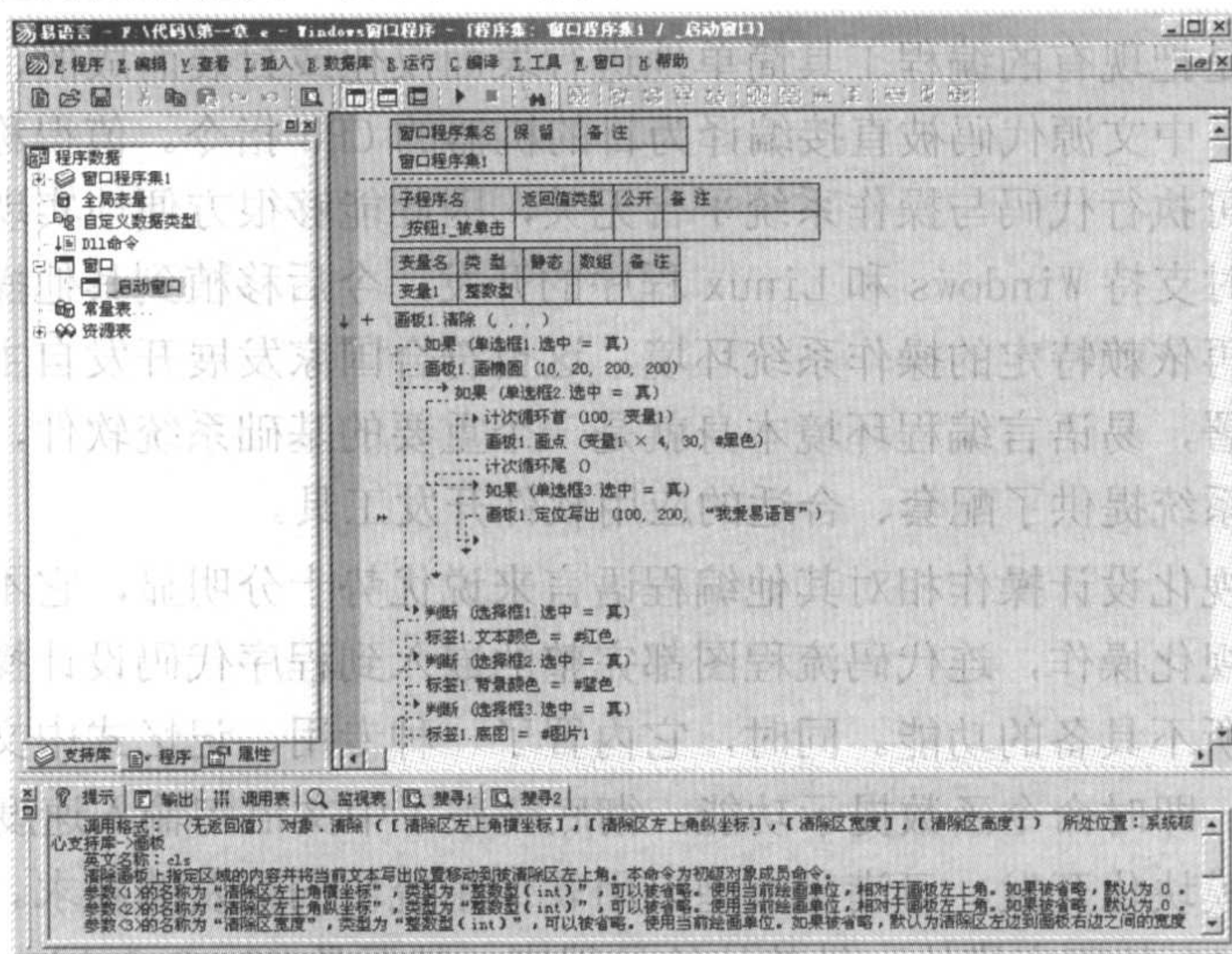


图 1-1 流程的即时可视化

### 2. 全中文

作为一款由中国人自己开发的编程语言，易语言在中文处理方面有良好的支持。用户在编写程序的过程中，可以不接触任何英文。根本不懂英文的人使用中文编写代码没有任何障碍。



(1) 中文名称的快速录入。易语言内置四种名称输入法：首拼、全拼、双拼、英文。三种拼音输入法均全面支持南方音。使用这些输入法能够极大地提高中文代码的输入速度。

直接使用系统提供的输入法，如五笔字型、智能 ABC、紫光拼音、自然码等，同样可以进行程序的输入。

(2) 程序全部以中文方式显示，运算符号全部显示为对应的中文符号 ( $\geq \leq \neq \approx \times \div$ )，日期时间以中文格式呈现 (年月日时分秒)，以便于中文用户理解、阅读程序。

(3) 适合中国人的语言、思维习惯。对其他计算机编程语言的学习，总会感到某种限制，首先是语言环境的限制，有很多专业的术语字面上很难理解它的含义，而以中文编写出的程序代码，符合中国人的语法习惯和逻辑思维，可以做到见文思义，更加适合中国人使用。在以后复查程序时可以非常直观地分析；给其他人源代码学习时也会非常简单，相互交流变得更加容易。

易语言更提供了中文格式日期时间处理、汉字发音处理、全半角字符处理、人民币金额处理等功能支持。

### 3. 全编译与跨平台

易语言拥有自己独立的高质量编译器，中文源代码被直接编译为目的机器的 CPU 指令，高效且不存在任何速度瓶颈和安全隐患。

易语言现已同时支持 Windows 和 Linux 程序开发，不再依赖特定的操作系统。

### 4. 可扩充支持库

易语言由基本系统和运行支持库两部分组成，两者之间通过使用易语言自行定义的接口技术进行协作。运行支持库内提供了易语言的所有语言要素，如：命令、窗口和报表单元数据类型、普通数据类型、常量等等。可以通过安装外部支持库来扩充易语言基本系统。运行支持库还可以被随意增减、抽换或升级，基本系统对运行支持库提供了详细的版本控制。本技术给用户带来的最大好处是：

(1) 用户可以根据行业或自身需要定制易语言；

(2) 由于运行支持库的不断增多、升级，易语言的功能将被迅速扩充；

(3) 由于运行支持库可以仅包含声明而不包含实际的运行支持代码，并且可以随时被更新或抽换，这样可使人们通过国际互联网与服务器进行远程易语言交流（譬如复杂型电子商务、远程控制等等）成为可能，这也是以后易语言互联网版本的发展方向。

### 5. 数据库支持

易语言相对其他编程语言的优势还在于易语言拥有自己的易数据库，并且用中文命令操作易数据库，简单方便。同时，易语言对外部数据库也有着非常好的支持，通过简单的组件和命令就可以实现易语言与各类数据库的连接，如 Oracle、MySQL、SQL Server、Access 等等。

### 6. OCX 组件、类型库 (TypeLib)、API 与 COM 对象

易语言可直接在程序中引用多种现有编程资源，极大的扩充了易语言的功能，并可对这些英文资源进行汉化处理，从而能够保持全中文的特点，让用户不用学习英文也能充分使用这些英文资源。

### 7. 与其他编程语言相互融合、互相补充

易语言支持当今先进的编程理念，譬如面向对象编程、事件消息处理机制等，了解、





学习和掌握易语言对掌握其他编程语言具有桥梁作用，同时，易语言可以和其他编程语言以标准 Win32 DLL 方式互相调用，保障了多种编程语言协同开发的需要。

### 8. 即时编译并自动规范语句格式的录入方式

在输入程序过程中，每条程序语句录入后，当光标离开该行，则对该行立即进行初步分析编译。如果该行输入正确，则该行的拼音简写会变成对应的汉字变量名或组件名，并呈现统一的字体间距和格式，因此任何人所编写的任何程序其格式都完全一致，这对于应用程序的协作开发、交流和维护非常有利。

### 9. 系统内置的自动名称管理器能够对用户所定义的各类名称进行跟踪管理

譬如：假设程序中现存在一个名为“刷新内容”的子程序，而且在很多地方都调用了该子程序。现在用户根据需把该子程序更改为另外一个名称，在传统的编程语言中，用户更改子程序名称后，要搜寻整个应用程序，逐一找到使用了该子程序的地方，把名称相应地改变过来。在易语言中，用户只需更改该子程序名称，程序中其他所有使用了该子程序的地方，其名称都将被自动更改过来。

### 10. 贯穿全程的即时且全面的信息帮助

用户在进行任何操作的过程中，随时按 F1 帮助键，均能够在状态行上或提示夹中获得有关当前操作位置的详细相关信息。譬如：用户将光标移动到某程序行上，然后按下 F1 键，马上就能够得到此程序行上所有命令的定义、参数、使用方法、所隶属的支持库等信息。

## 1.2 易语言的安装

易语言安装很简单，和很多软件的安装类似，安装过程中没有复杂的选项。

首先，从光盘或下载文件目录中找到易语言的安装文件并双击运行安装程序。如图 1-2 所示。

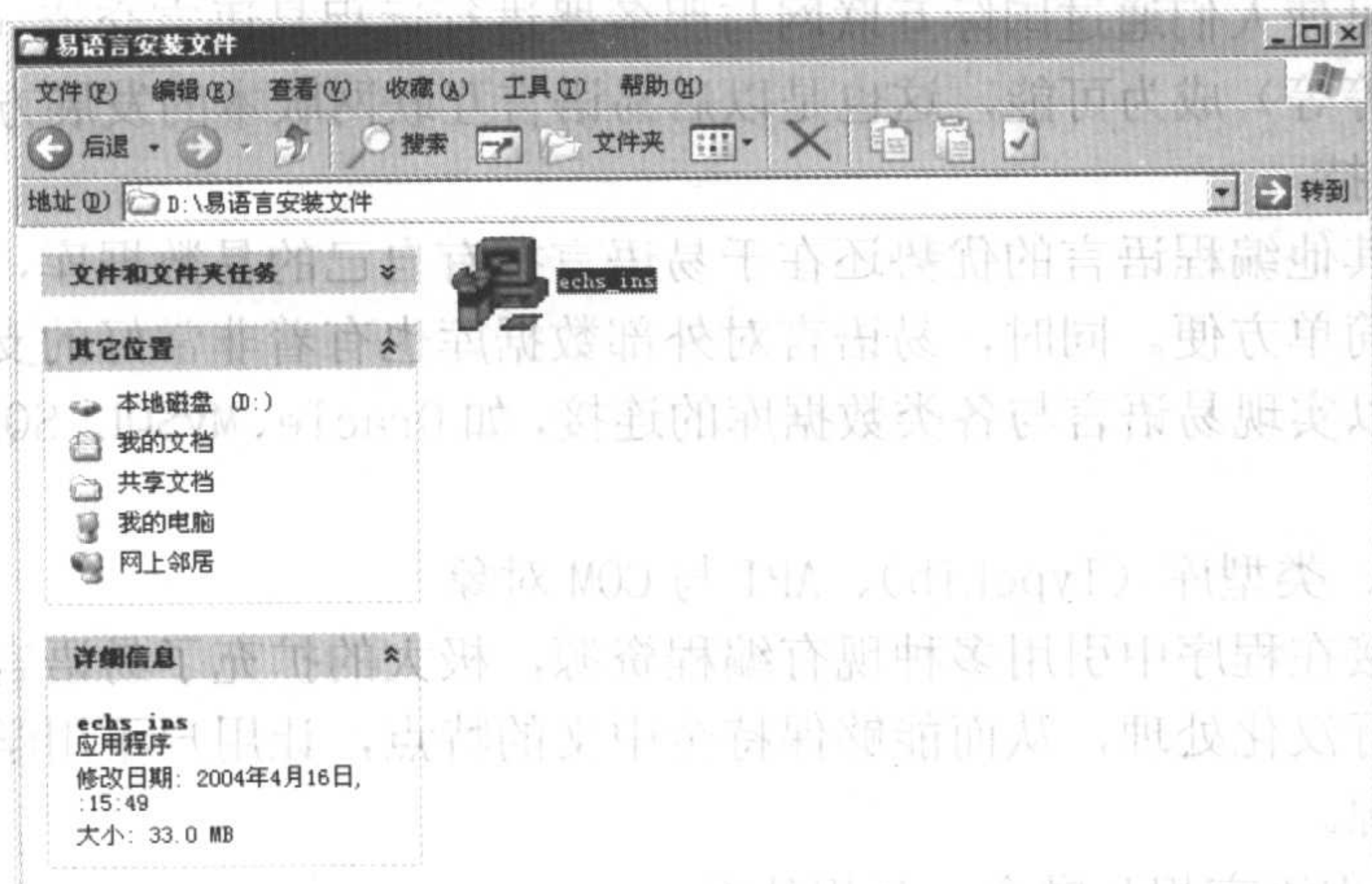


图 1-2 找到易语言安装文件



然后在弹出的安装界面中点击“下一步”按钮。如图 1-3 所示。

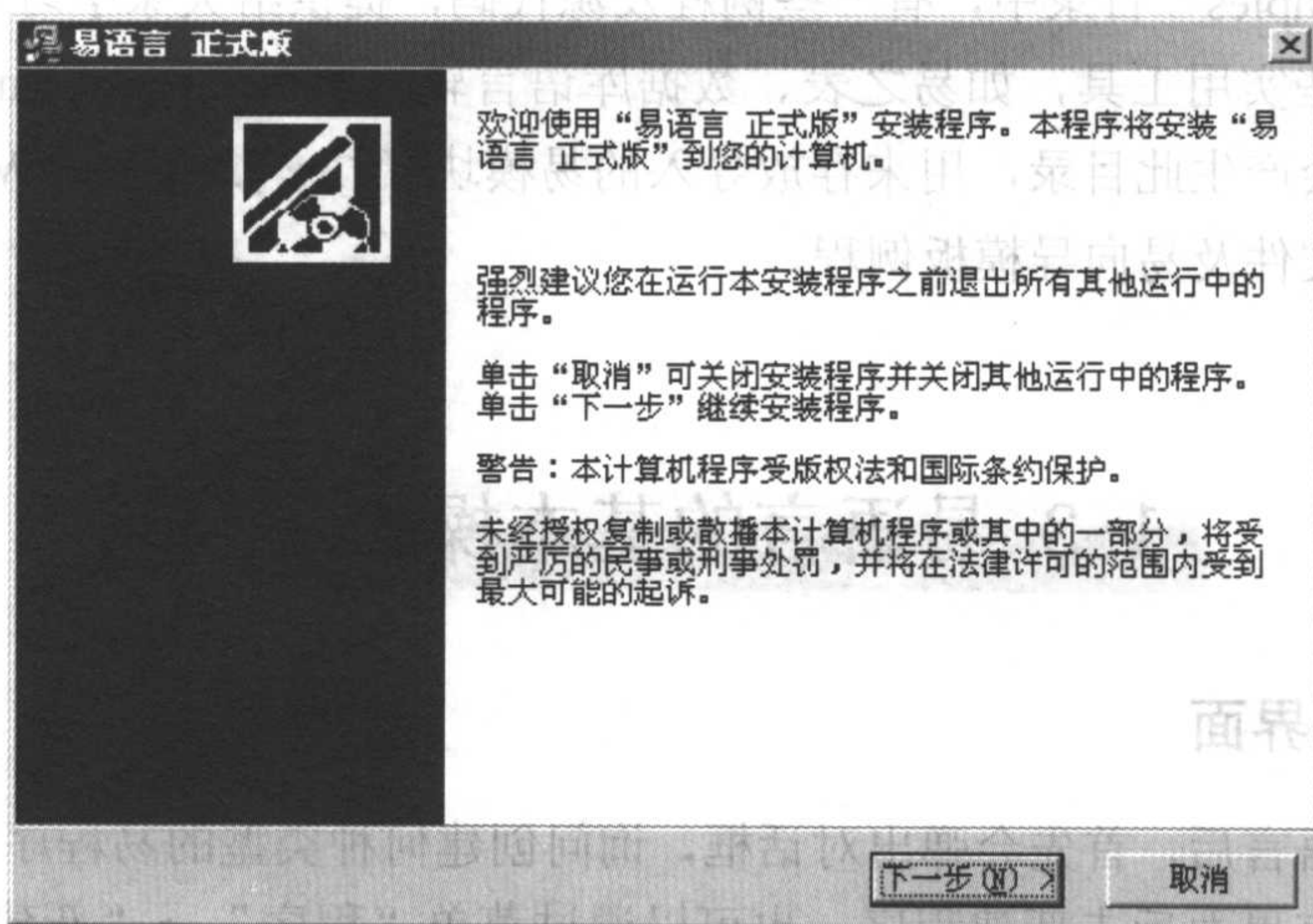


图 1-3 易语言欢迎界面

点击“下一步”后会出现“许可协议”和“自述文件”，依次点击“是”和“下一步”，接下来出现的窗口都点击“下一步”按钮，然后等待易语言的安装。如图 1-4 所示。

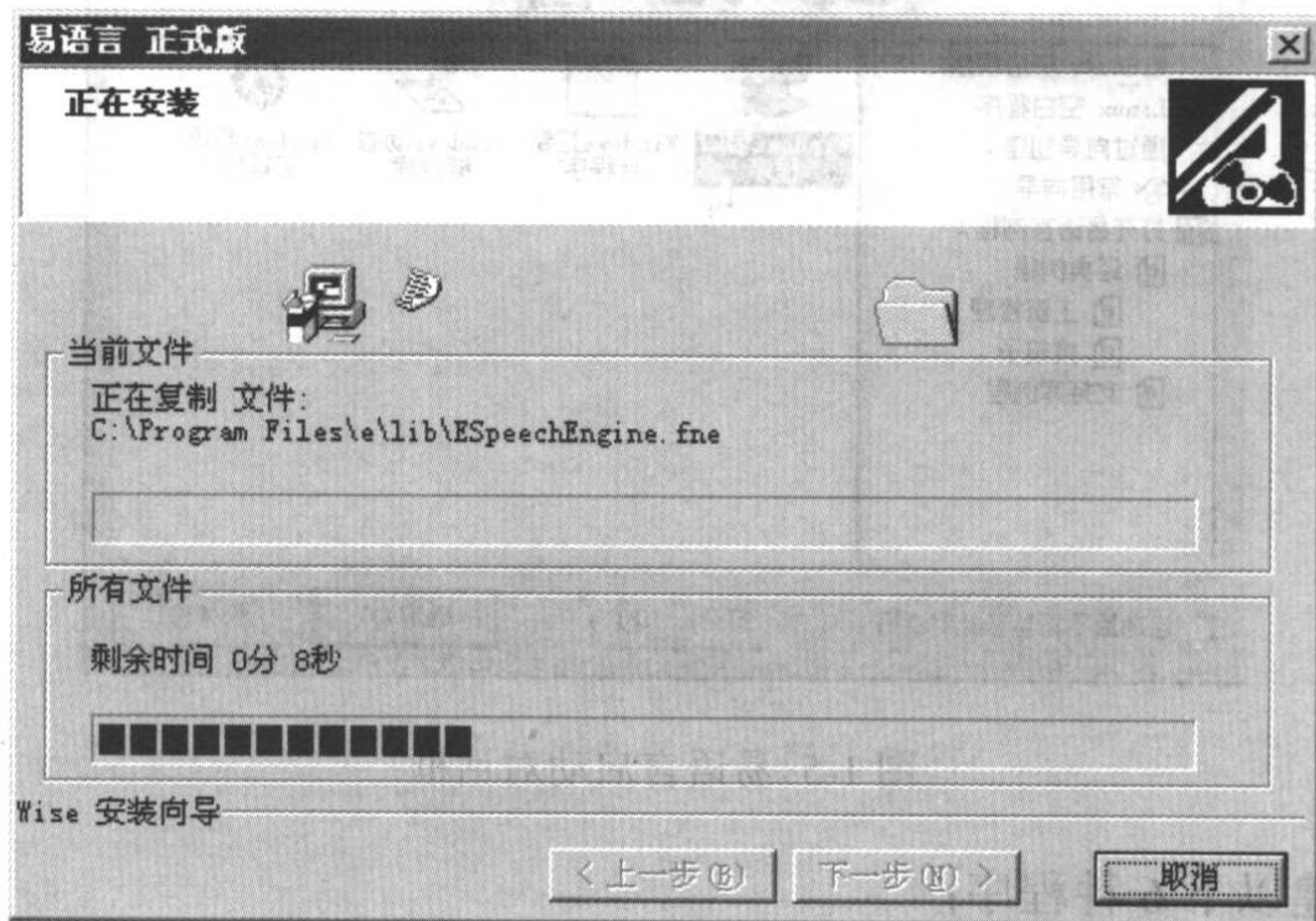


图 1-4 易语言正在安装

最后，点击“完成”按钮，易语言就安装好了，在桌面上和开始菜单中会出现易语言的快捷方式，点击快捷方式即可运行易语言。

易语言的卸载可以通过运行易语言安装目录中的 UNWISE.EXE 文件，或在控制面板中的“添加或删除程序”中进行卸载。

易语言安装后，可以到易语言的安装目录中，了解一下安装目录中各子目录的作用。“clr”目录中，存放着易语言的各种颜色配置文件；“help”目录中，存放易语言的帮助文





档;“lib”目录,存放易语言的基本支持库和扩展支持库等重要文件,不要随意移动或删除其中的内容;“samples”目录中,有一些例程及源代码,提供给大家学习参考;“tools”目录中,提供了一些实用工具,如易之表、数据库语言转换器等等;“ecom”目录,在导入过易模块以后,会产生此目录,用来存放导入的易模块(\*.ec 文件);“Wizard”目录,存放易向导可执行文件及易向导模板例程。

### 1.3 易语言的基本操作界面

#### 1.3.1 易语言的界面

初次运行易语言后,首先会弹出对话框,询问创建何种类型的易程序。如图 1-5 所示。

若打开易语言界面后未新建程序,也可以通过菜单“程序”→“新建”来创建新的易程序。或点击窗口工具条中的新建按钮来新建易程序。

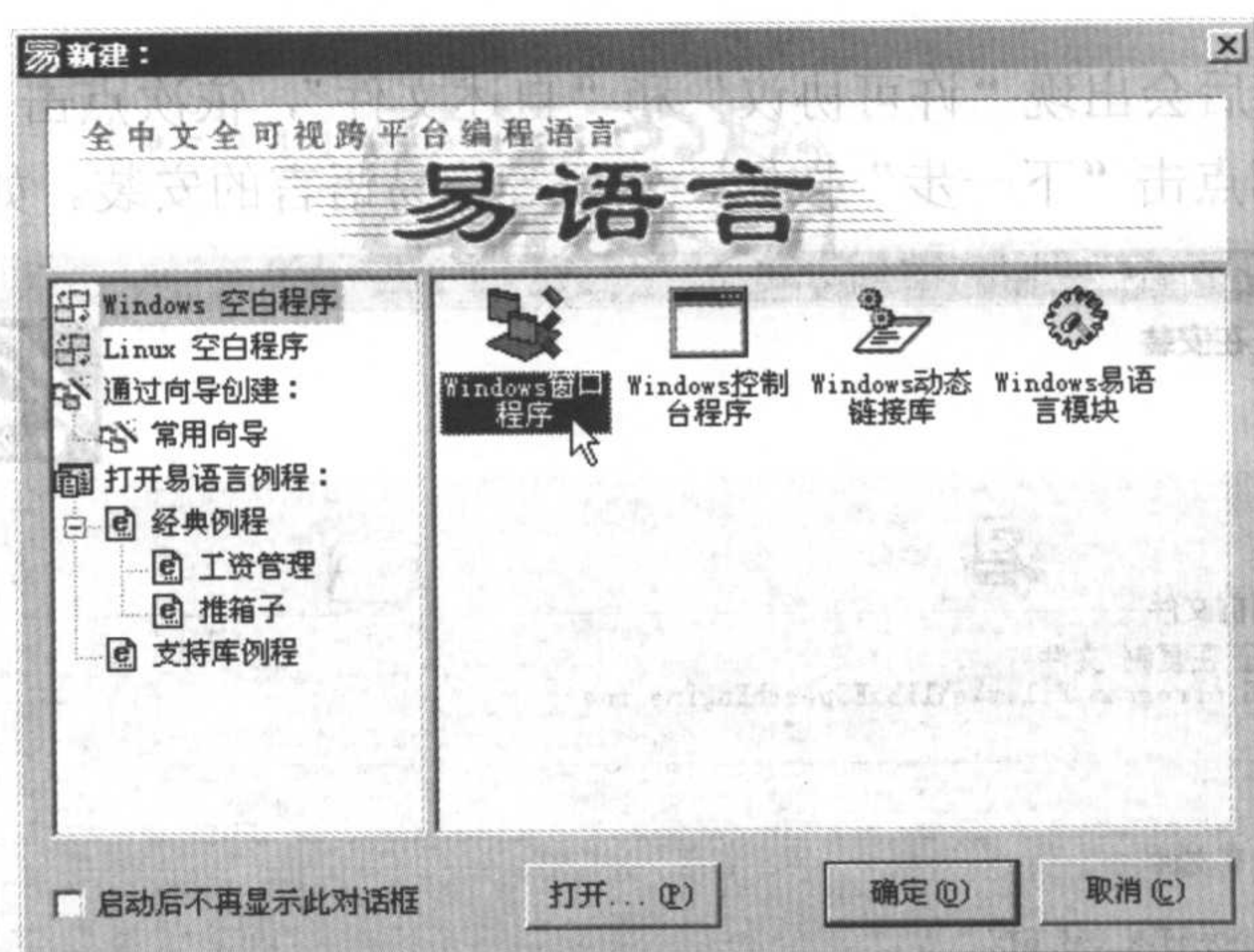


图 1-5 易语言启动对话框

易语言可创建以下 6 种程序:

1. **Windows 窗口程序:** 是支持在 Windows 下弹出窗口及组件等标准 WIN32 位程序,也称易程序。
2. **Windows 控制台程序:** 是 WIN32 位无窗口界面的易程序。一般用于服务器等。
3. **Windows 动态链接库:** 可以生成 DLL 程序,将在本书关于 DLL 章节中介绍。
4. **Windows 易语言模块:** 简称易模块,是经过初步编译后的程序模块,供其他程序重复调用,将在本书关于模块章节中介绍。
5. **Linux 控制台程序:** 是支持 Linux 操作系统的无窗口命令行程序,将在本书关于 Linux 程序章节中介绍。
6. **Linux 易语言模块:** 是支持 Linux 操作系统且经过初步编译后的程序模块。



选择“Windows 窗口程序”，点击“确定”按钮，就会创建一个相应的标准的 Windows 窗口程序，并可以看到易语言的主界面。

易语言主界面的最上方是标题栏，显示易语言版本及当前打开的程序名称，当前窗口名称，以及当前所支持的操作系统。标题栏下方是菜单栏，有易语言的常用菜单。菜单栏下方是快捷命令按钮工具条，一些常用的操作都可以通过点击这些工具条中的按钮实现。

主界面的左边是易语言的工作夹，其中有 3 个面板，分别是“支持库面板”、“程序面板”和“属性面板”。

“支持库面板”的作用是：显示支持库列表，展开查看各支持库提供的命令、数据类型等信息。在程序编辑状态下，可以通过双击此面板中的某个命令，将其直接填充到光标处。若有窗口组件的方法也可以在这个列表中查看方法的用处。将光标移至某支持库根部，按下 F1 后可查看此支持库的介绍信息。

“程序面板”的作用是：相当于一个组织机构，可以添加窗口，或加载全局变量、常量、资源、DLL 命令申明、自定义数据类型等。也可用来在程序各操作界面间进行切换，例如可以直接找到某个创建的窗口中，或快速找到某个子程序，

“属性面板”的作用是：属性表可查看和更改已添加组件的属性、组件列表列出所有组件并可快速选择所需组件，事件列表可生成此组件的事件子程序。

最右边是易语言的组件箱，里面列出了易语言提供的所有组件。分为四栏，“基本组件”栏可显示易语言最基本常用的组件，即核心支持库内的组件，在本书基本组件章节中进行介绍。“扩展组件”包含扩展支持库内的组件，在本书的后面有一些介绍。“外部组件”包含 COM 包装支持库所封装的 ActiveX 组件，此组件也称 OCX 组件，将在外部 OCX 组件章节中进行介绍。“外部事件组件”包含 COM 包装支持库所封装的 COM 事件组件。

主界面中间是设计区，在窗口设计时可自由向窗口中添加组件，进行程序界面设计；在程序代码编辑状态下可录入、修改程序代码。切换这两个工作状态可通过“窗口”菜单或“程序面板”等实现。

最下方是易语言的状态夹，可以查看帮助信息，查看调试文本等等。调试在本书附录中进行介绍。如图 1-6 所示。

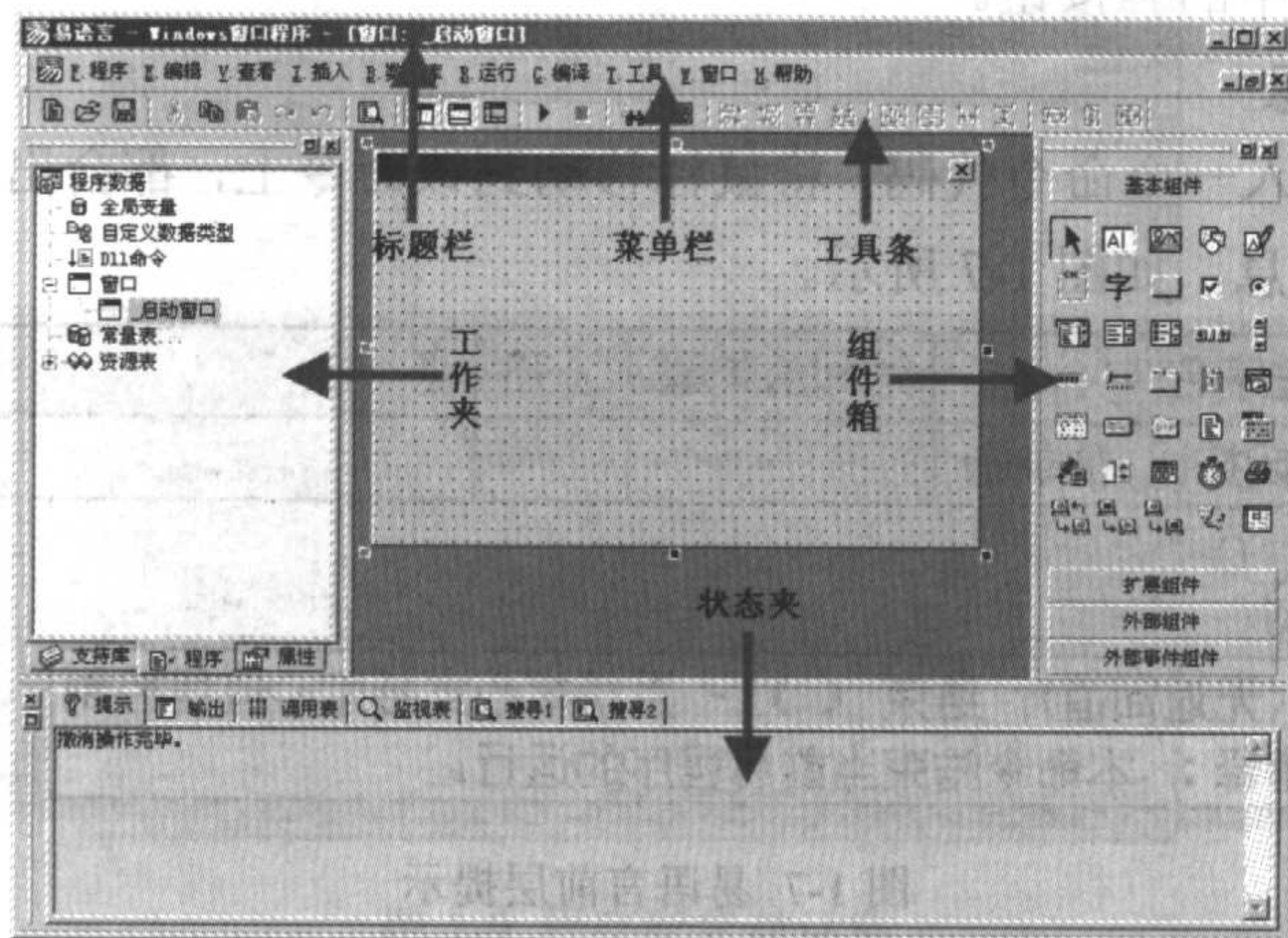


图 1-6 易语言主界面





## 1.3.2 如何在设计窗口中添加组件

从组件框中选出所需的组件添加在设计窗口中，只需要用鼠标左键在组件箱中点击欲添加的组件，使其处于选中状态，然后在设计窗口中按住鼠标左键拖动，拉出一个该组件即可。添加后的组件可以通过拖动鼠标改变其位置和大小，也可以使用方向键来微调组件的位置，还可以按住 Shift 键+方向键来微调组件的大小。

## 1.3.3 如何使用易语言帮助系统

易语言的帮助系统分为“即时帮助信息”和“易语言知识库”。

### 1. 即时帮助信息

易语言编程环境在用户进行任何操作的同时，会将有关的支持信息在提示面板中显示出来，可以使用以下介绍的方法来查看即时帮助信息：

随时按下“F1”热键使用可随时得到与主题相关的帮助信息。

使用菜单：“帮助”→“即时帮助”可得到与主题相关的帮助。即时帮助信息内容是在用户进行任何操作的同时，将有关的支持信息在提示面板中显示出来。

即时帮助信息可显示系统中各运行支持库内的命令、库定义数据类型、库定义常量等信息。直接在工作夹内的支持库面板中找到并单击欲查找信息的项目，此时所有的相关信息将会显示在状态夹的提示面板中。

如果欲将这些信息提取出来打印或者以后阅读，可以在相应项目上单击鼠标右键，在弹出菜单中选择“拷贝帮助文本到剪贴板”或者“写帮助文本到文件”，输出与该项目及该项目所有子项目相关的帮助信息，供电脑中浏览或打印出来阅读。

### 2. 易语言知识库

易语言的帮助文档已经相对成熟了，包含了当前易语言版本的所有帮助信息，以及大量的贴图和源代码，为学习易语言提供了很大帮助。

打开易语言知识库可以通过点击易语言“帮助”菜单中的“易语言知识库”选项或直接点击易语言工具条中的按钮。

### 3. 前层提示信息

易语言中，每输入一个命令代码，将鼠标移动到该命令上，都会出现一个信息提示框，显示该命令的帮助信息。如图 1-7 所示。

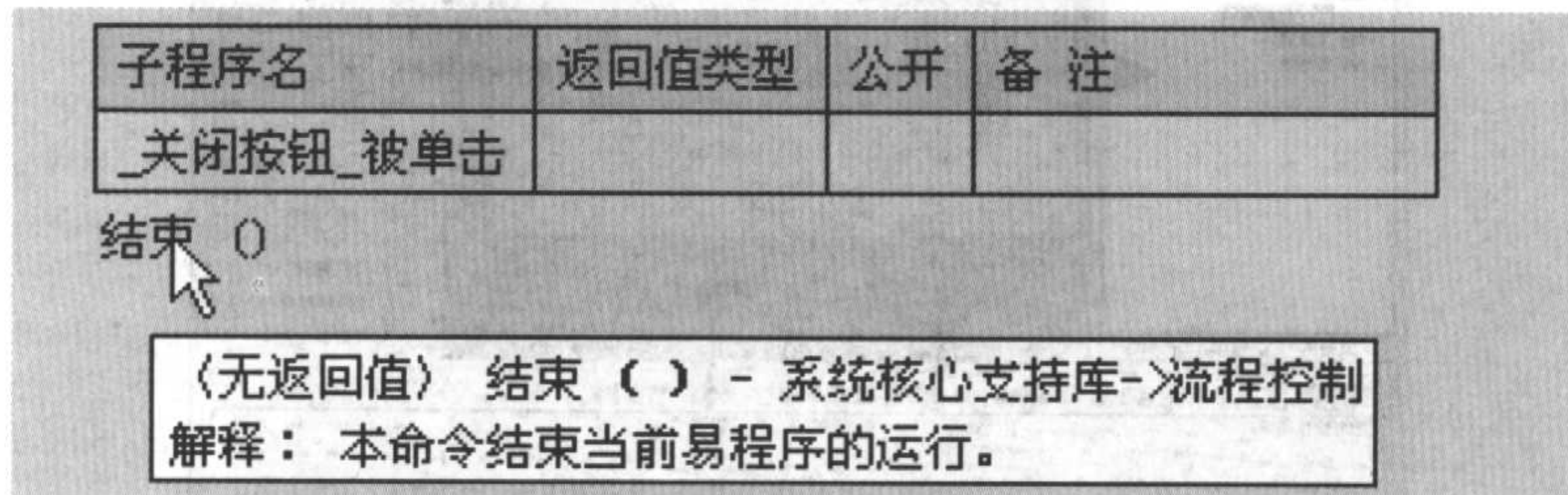


图 1-7 易语言前层提示



### 1.3.4 如何配置易语言

可以根据个人习惯，进行以下设置：系统配置，程序配置，支持库配置等。

#### 1. 系统配置

点击菜单“工具”→“系统配置”。可以打开易语言的系统配置对话框，通过调整该对话框中各项属性的参数，可以自定义界面各部位颜色，可以选择各种配色方案，还可以更改代码字体，和对内置输入法等很多方面进行配置。如图 1-8 所示。

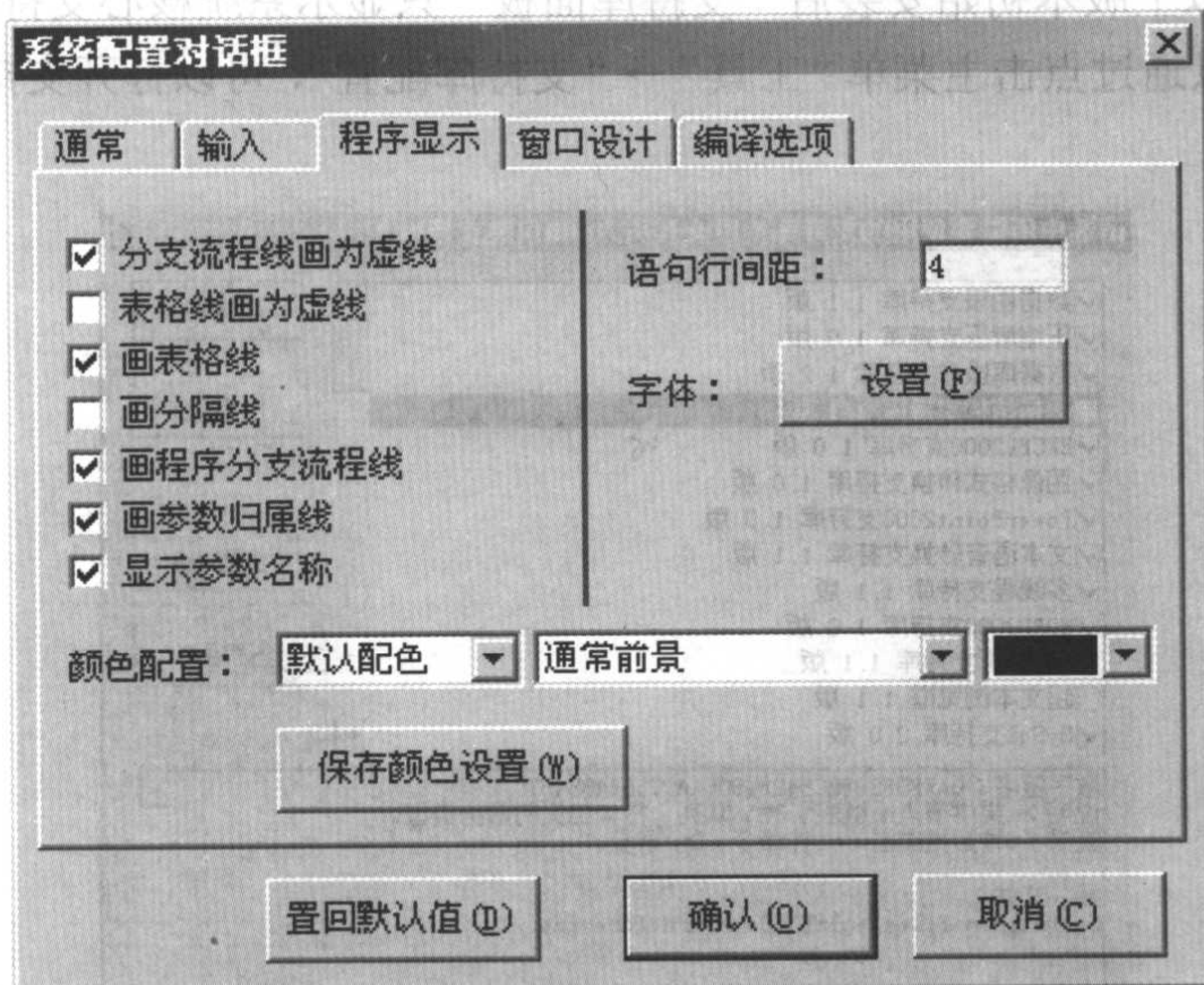


图 1-8 系统配置对话框

#### 2. 程序配置

点击菜单“程序”→“配置”，可以打开程序配置对话框。如图 1-9 所示。

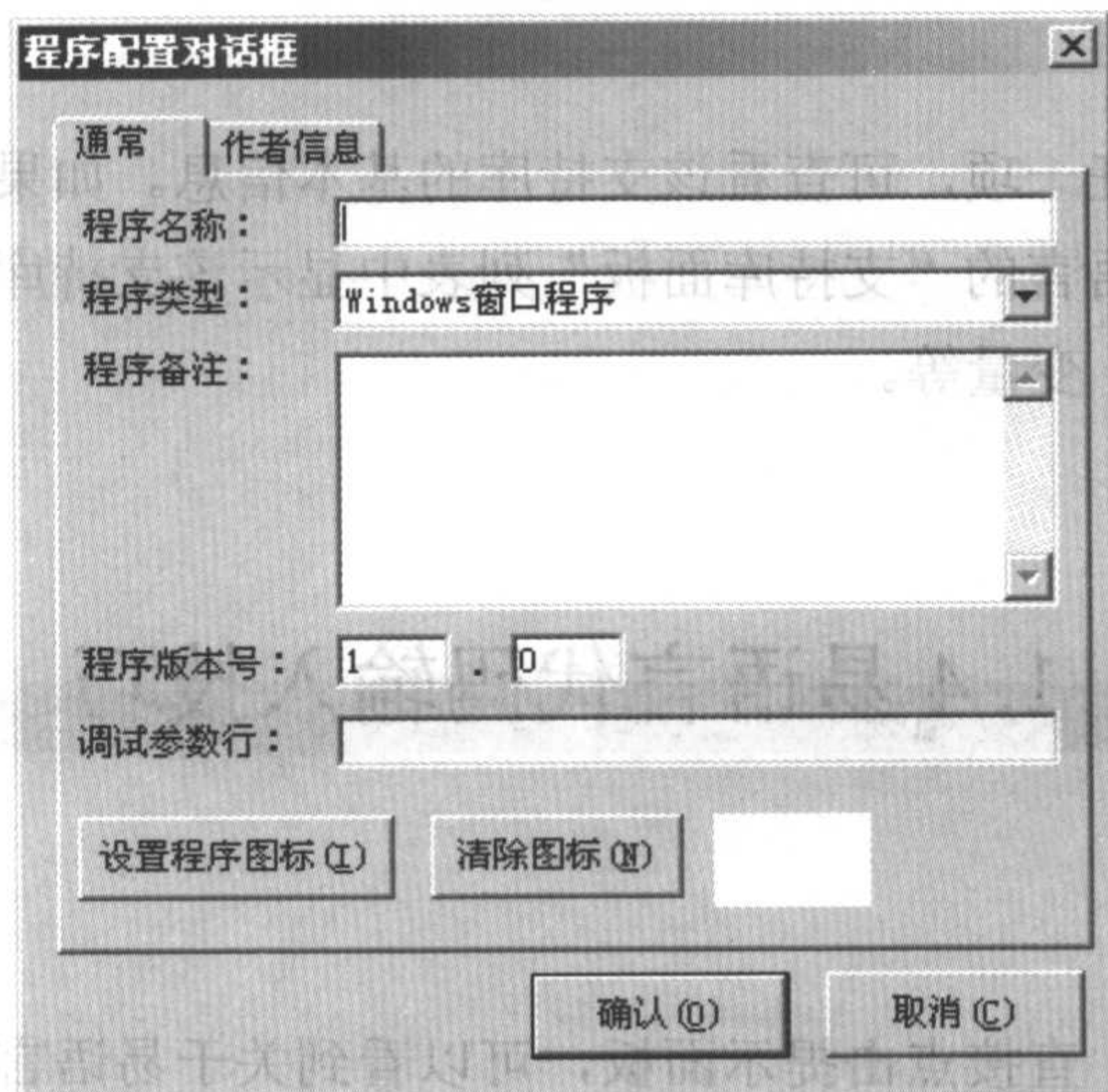


图 1-9 程序配置对话框





该对话框可以将程序名称、程序备注、作者信息等信息保存在生成后的 EXE 文件中，当查看此 EXE 文件的属性时，这些信息会显示出来。并且可以在这里为程序设置图标。

如果每次要生成 EXE 文件，然后运行带参数的 EXE 文件是非常麻烦的，在本对话框中输入“调试参数行”后，当调试运行该程序时，易语言将自动附加该参数。生成 EXE 文件后，此调试参数行内容不会保存至程序中。

### 3. 支持库配置

易语言 3.8 以上版本初始安装后“支持库面板”只显示系统核心支持库，如果使用到更多的支持库可以通过点击主菜单“工具”→“支持库配置”，可以打开支持库配置对话框。如图 1-10 所示。

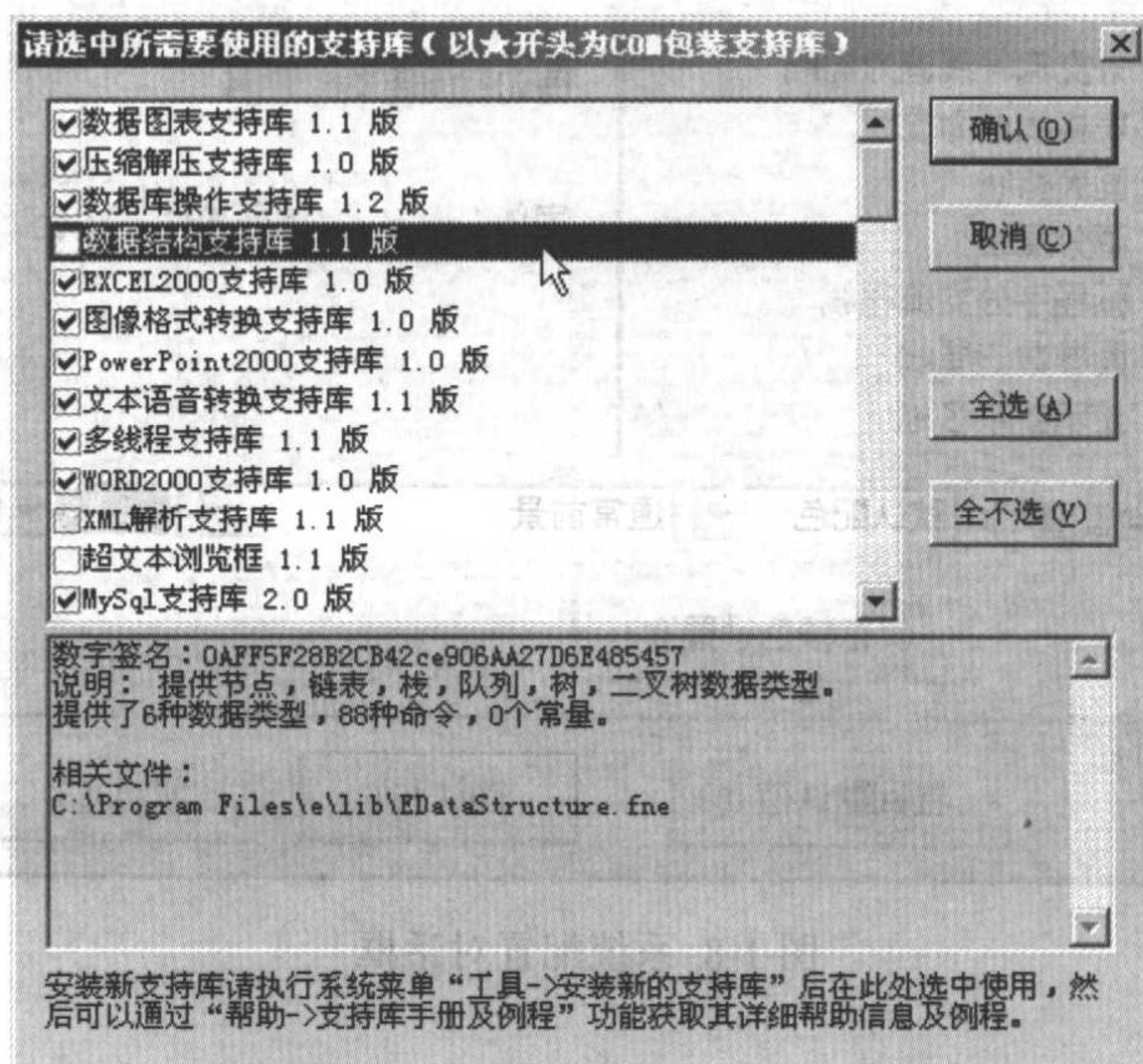


图 1-10 支持库配置对话框

点击支持库列表中任一项，可查看该支持库的基本信息。如果将某支持库名称前的对钩“√”加上，则在易语言的“支持库面板”列表中显示该支持库，并可以程序中使用其提供的命令、数据类型、变量等。

## 1.4 易语言代码输入技巧

### 1.4.1 内置输入法

在易语言刚运行时，直接点击提示面板，可以看到关于易语言程序输入方法的详细提示信息。如图 1-11 所示。



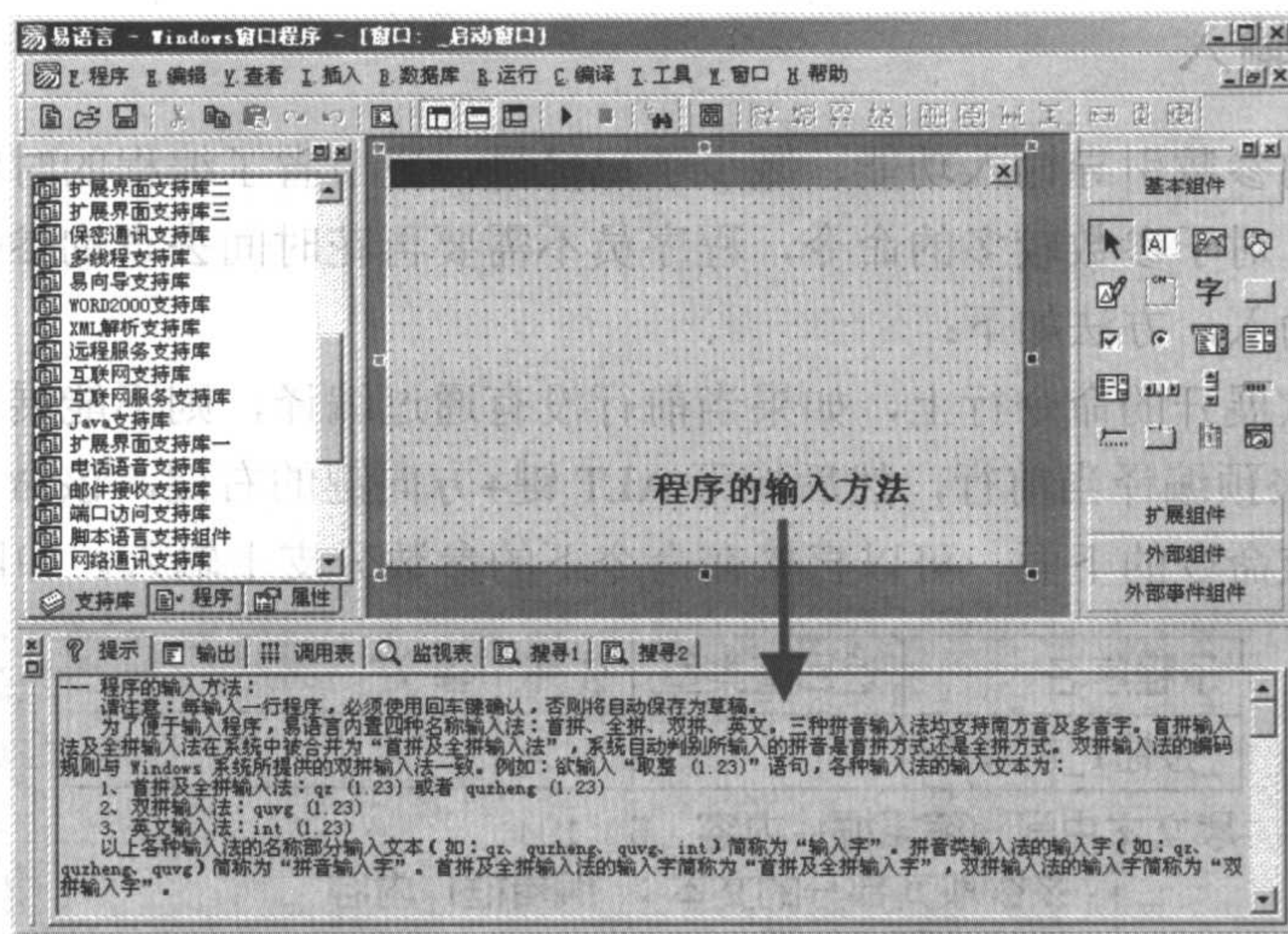


图 1-11 程序输入方法提示

易语言内置四种输入法：首拼、全拼、双拼、英文。三种拼音输入法均支持南方音及多音字。首拼输入法及全拼输入法在系统中被合并为“首拼及全拼输入法”，系统自动判别所输入的拼音是首拼方式还是全拼方式。双拼输入法的编码规则与 Windows 系统所提供的双拼输入法一致。例如代码：求正弦(20)，各种输入法的输入文本如下：

首拼输入法：

qzx (20)

全拼输入法：

qiuzhengxian (20)

双拼输入法：

qqvgxm (20)

英文输入法：

sin (20)

代码中涉及的汉字，都可以使用这几种内置输入法输入。

一些特殊情况：

对于没有声母的汉字（如“按”），使用首拼输入法时应取其韵母全部字符。比如：“按钮”，用首拼输入法就可以输入成“ann”（其中“an”为“按”的韵母，“n”为“钮”的声母）。

如果要输入的名称中既有汉字又有英文字母，则其中的英文字母不论大小写都要用大写英文字母输入。例如：要输入“编辑框 x”，使用首拼输入法就要输入：“bjkX”。

### 1.4.2 系统输入法

使用五笔字型、自然码、智能 ABC 等这些由 Windows 提供的系统输入法，在易语言中也可以进行程序代码的输入。在输入代码时打开该输入法即可。如果以前五笔字型很熟练的话，也可以很快的输入代码。





### 1.4.3 参数分步输入

易语言提供的参数引导输入功能，减少了记忆量，更节省了编程的时间，极大降低了程序录入的错误。对于参数较多的命令，程序员不需要再花时间去查询参数的意义，可以直接将命令展开输入，方法如下：

将光标停在欲展开的命令行上，如果当前行没有通过编译，则不能展开命令，可以使用 Shift+Enter 键来预编译当前行，然后按下 ALT 键+方向键的右键，该命令就会被展开，各参数都列在了该命令的下面，可以直接在命令下的参数分支上输入。如图 1-12 所示。

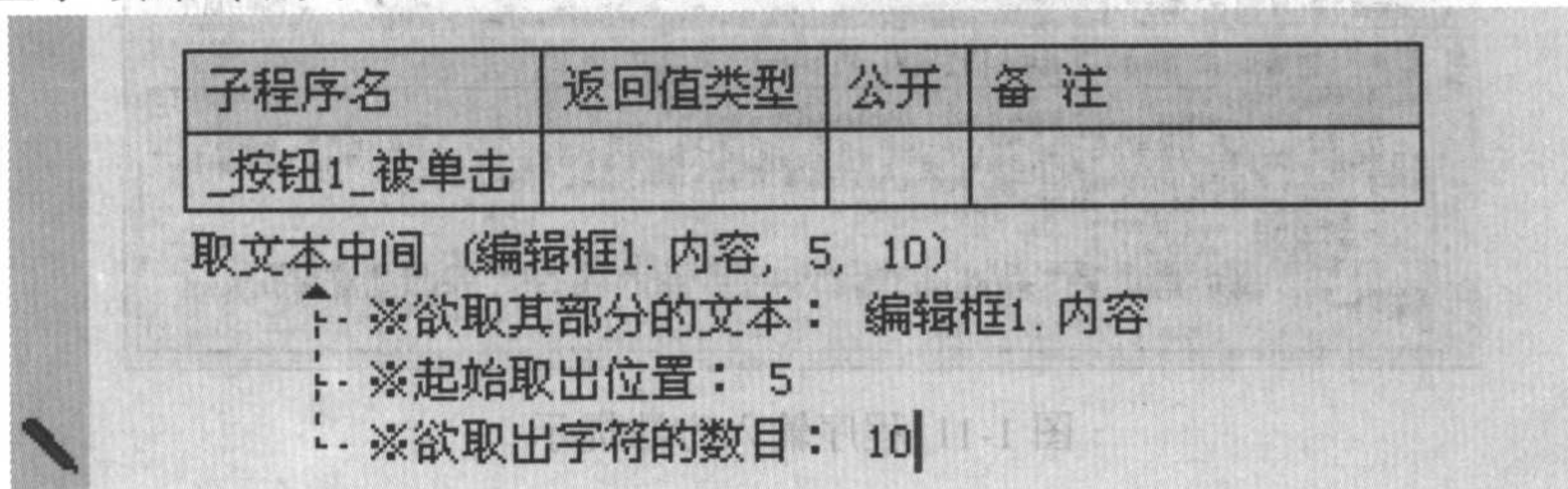


图 1-12 分步输入参数

### 1.4.4 输入备注与代码屏蔽

#### 1. 备注输入

备注是一行或一段代码的提示和说明。编写代码时一定要养成一个良好的习惯，就是给部分代码输入备注信息，这样一来，既方便了自己日后阅读代码，又方便其他人更快的理解程序代码的思路和功能。

输入方法：在备注文字前加 “'” 号，则该符号后的本行文字变为备注，在输入代码时，可以在代码的旁边或代码的下方输入备注。如图 1-13 所示。

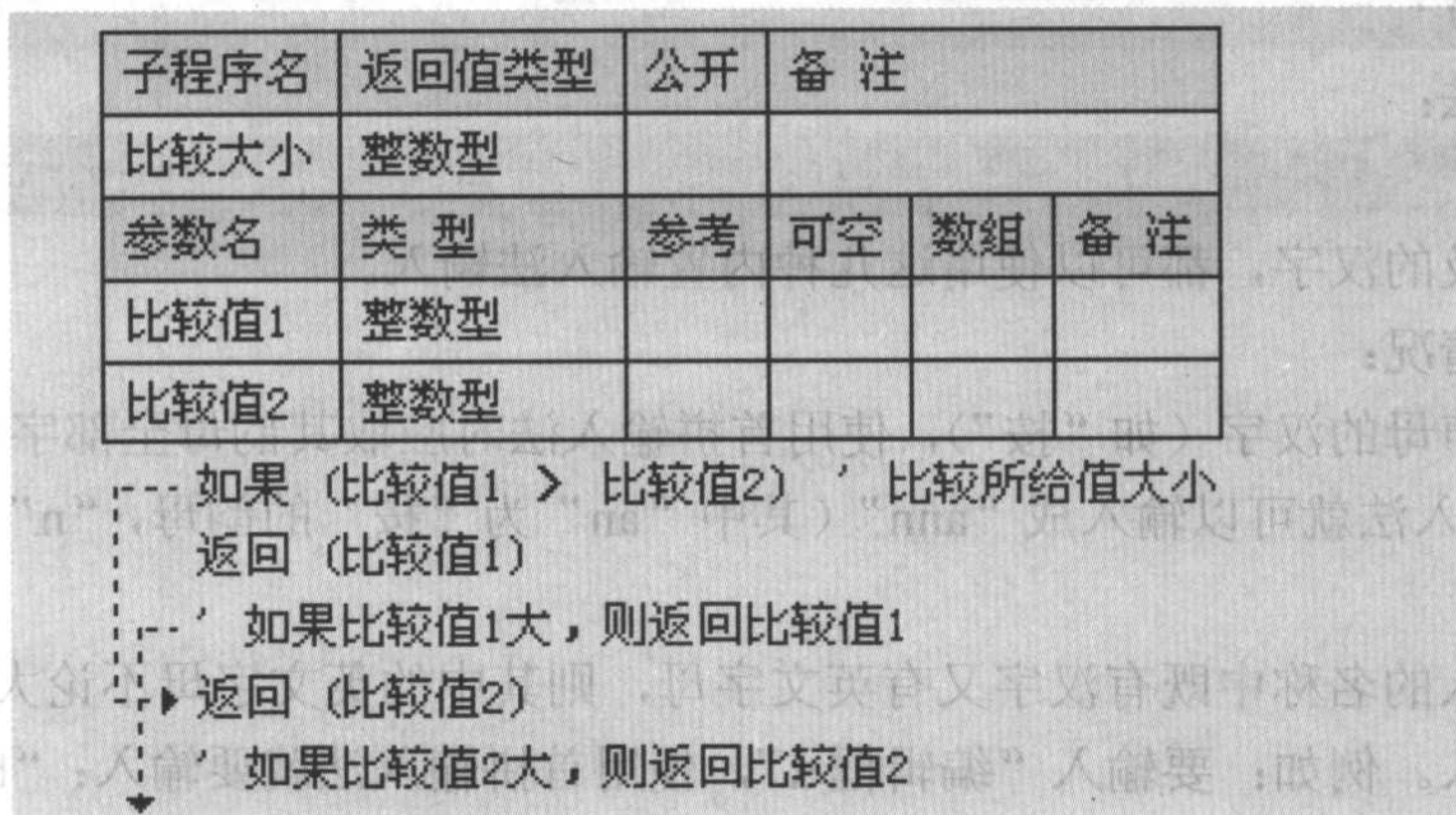


图 1-13 输入备注

#### 2. 屏蔽与批量屏蔽

在任何情况下，如果想屏蔽一行代码，则在该行代码前加 “'” 号，和置为备注的方法相同，屏蔽后的代码在运行调试时不会被编译，调试程序寻找错误时，该方法会起到很大



作用。将代码前的“'”号删除便可以解除屏蔽。

还可以在代码上点击鼠标右键，弹出的菜单中也有“屏蔽”和“解除屏蔽”选项。屏蔽的快捷键是 Ctrl+K 键，可以选中多行代码，然后使用 Ctrl+K 键来屏蔽多行代码，然后可以使用 Ctrl+M 键来解除屏蔽。

#### 1.4.5 四种输入语句分类

易语言常见的程序语句分为四种：赋值型语句、非运行语句、方法语句、命令语句。

1. 赋值型语句，即给某组件属性或某变量赋值的语句。一般使用“=”连接被赋值方和赋予的值，并且赋予的值一定要和被赋值的属性或变量的数据类型相同或互相兼容。例如：

(1) 给组件属性赋值。

标签 1. 标题 = “全中文易语言”

(2) 给变量赋值。

整数型变量 1 = 29000

2. 非运行语句，即在运行过程中不被运行的语句。包括注释型语句和草稿型语句。

3. 命令语句，即执行命令使用的语句。命令是一种程序运行动作指令，易语言的命令由易语言基本支持库和扩展支持库提供，命令的调用格式为：命令名（参数，...），例如：

信息框（“大家好！”， 0，）

其中括号前是命令名，括号中的内容是命令的参数，主要是供命令进行判断、选择或再加工的因素，每个参数用逗号分隔。

4. 方法语句，方法是一个具体对象能够执行的动作，方法类语句就是执行某对象的方法使用的语句，对象中的方法是通过发送消息实现。方法的使用格式和命令类似，很多方法都有参数表，调用格式为：对象.方法名(参数 1，参数 2，...)。一般方法都是指组件的方法。例如：

画板 1. 滚动写行（“您好，祖国！”）

#### 1.4.6 易语言中的关键字

易语言中所有的命令名、组件的属性名都可被看做是易语言的关键字。易语言中的组件名称、变量名称和子程序名称等等都是可以自定义的，所以在起名称的时候既要清楚明确又要防止和这些关键字重名。

虽然是可以利用这些关键字来起名，在有重名的时候系统也会自动提示，但为了减少不必要的麻烦，还是要尽量避免重名。





## 1.5 第一个易程序

### 1.5.1 易程序的结构

下面对易程序的结构进行介绍。首先您的易程序需要有一个显示界面，一般是使用一个窗口作为启动画面，易语言中指定“\_启动窗口”这个窗口是首先弹出的窗口，大家可以在这个窗口中放上其他的组件，以显示信息或美化程序界面。窗口显示时会有一系列的触发事件，如“创建完毕”事件、“尺寸被改变”事件等，但可能大家没有用到这些事件，因此不会进行任何的动作，只是显示一个窗口。若大家使用到了这些事件，就会形成事件子程序，这样就产生了子程序，而子程序是放在程序集中进行组织的，而每一个窗口对应一个程序集，大家也可以自己创建自己的程序集，程序集包含若干个子程序，子程序内输入程序代码。而程序代码就是各种命令和方法。

为配合命令书写，需要有存放内容的变量，为方便引用，可以建立常量，自定义数据类型，甚至可以建立图片或声音资源供引用。为了调用系统应用程序接口 API，使用更多的功能，需要进行 DLL 声明。这些操作可以在“程序面板”中完成。

为了重复利用程序资源，不必每次都重新写某段代码，除提供自定义子程序外，还提供易模块，供其他程序调用，也可以写标准动态链接库，供易语言及其他语言调用。

为了理解上述的程序结构，下面跟着本书写第一个易程序，体会其中各部分的联系。

### 1.5.2 开始写第一个易程序

下面就来编写第一个易程序。本程序将在一个窗口中显示一个按钮，点击这个按钮后就会显示“祖国您好”这几个字。

为实现上述效果，必须有显示文字的地方。在此所有的显示载体都是由窗口内操作，使用标签显示文字，当然，还需要有一个按钮来接收用户鼠标的点击。下面跟着程序步骤建立第一个易程序。

首先新建一个易程序，“易程序”即新建窗口中的“Windows 窗口程序”。以后本书所有章节中所提到的“易程序”，都是指“Windows 窗口程序”。

然后在“\_启动窗口”中添加 1 个标签组件和一个按钮组件。如图 1-13 所示。

**注意：**打开菜单“工具”→“系统配置”中的“通常”标签，可以发现易语言可以通过两种方式启动程序：一种是通过名称为“\_启动窗口”的窗口启动，另一种是通过名称为“\_启动子程序”的子程序启动。默认情况下是第一项，即编译好的易程序运行时第一个运行“\_启动窗口”，因此不要改动这个窗口的名称，否则编译时会给出错误提示。

添加组件时，选用鼠标点击组件，移动鼠标至窗口中，再在窗口空白处象画一个矩形框一样，从左上角向右下解按下鼠标不松手，以拖出新组件的轮廓。



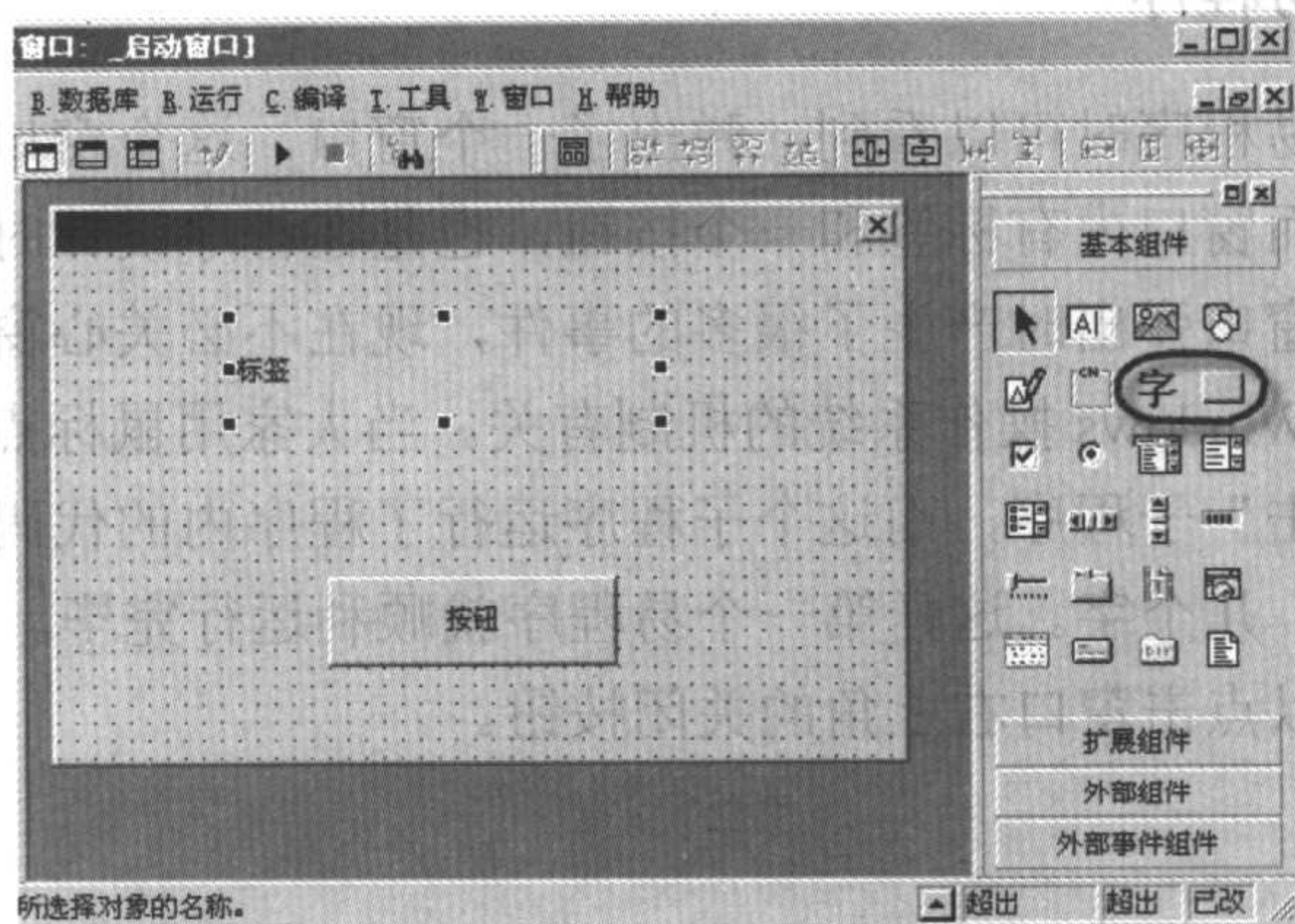


图 1-13 第一个程序界面设计

然后用鼠标双击窗口中的按钮组件，会自动切换到代码编辑界面，并自动生成“\_按钮1\_被单击”子程序。然后在光标处输入以下代码：如图 1-15 所示。

标签 1. 标题 = “祖国您好”

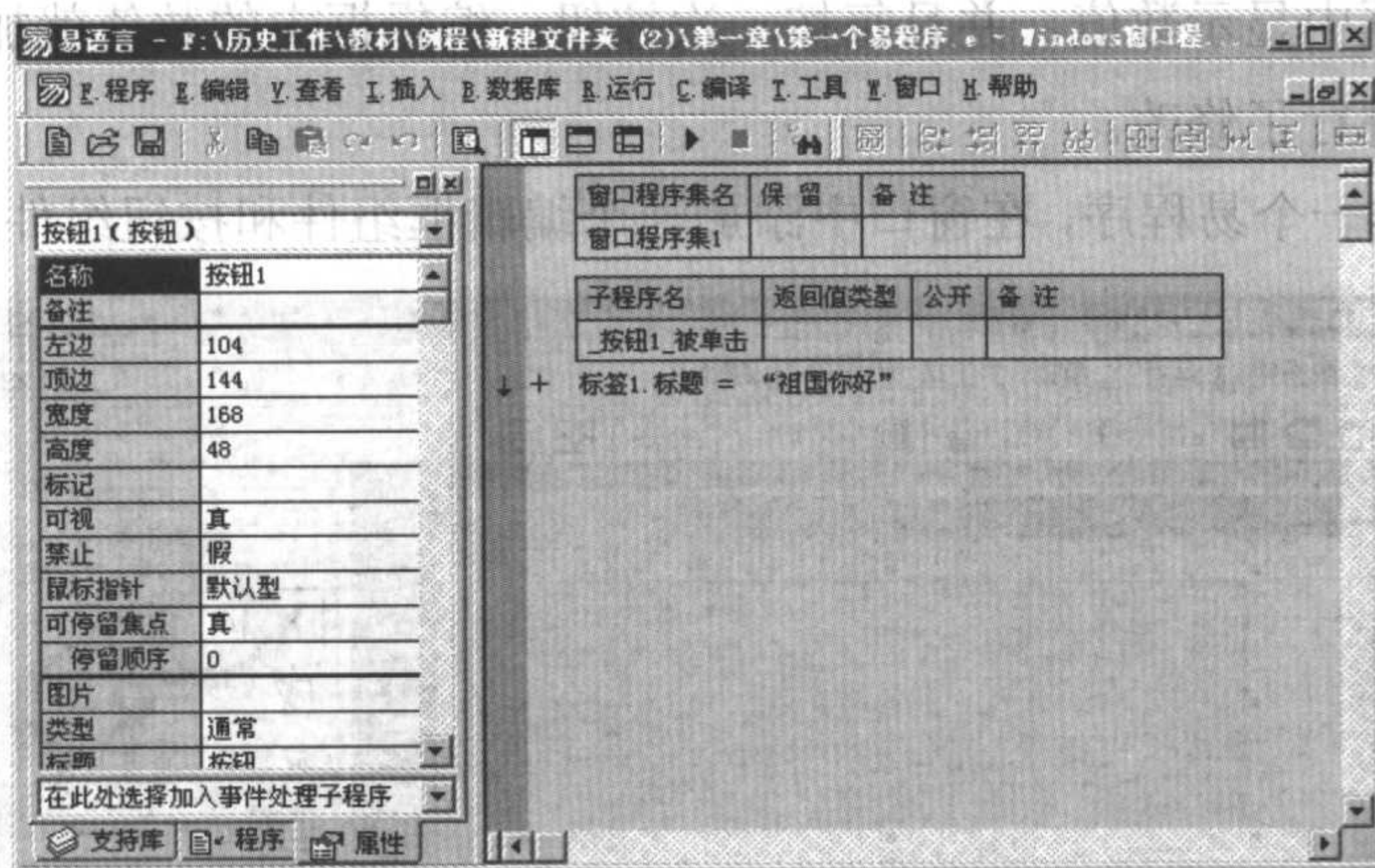


图 1-15 在“\_按钮 1\_被单击”子程序中输入代码

代码输入完毕后，就可以试运行程序了。可以点击易语言工具条中的运行按钮，也可以按下 F5 键来运行程序。程序运行后点击窗口中的按钮，标签显示出“祖国您好”。如图 1-16 所示。

最后可以通过“编译”菜单编译为 EXE 文件发布。有关编译的内容在本书的附录中。



图 1-16 第一个易程序运行效果





## 1.5.3 分析第一个易程序

大家运行第一个易程序时可以看到，弹出了一个窗口，这个窗口就是大家新建易程序时的“\_启动窗口”，而窗口内有标签和一个按钮，也是在程序设计时放上去的，现在清楚地显示出来。当然在窗口弹出时产生了很多的事件，现在不必关心窗口是如何发送消息，显示窗口的，这只与 Windows 操作系统的机制有关。当大家用鼠标点击按钮时，就运行了那个“\_按钮 1\_被单击”子程序，而这个子程序运行了程序内的代码，即改变了标签的标题，显示“祖国您好”几个字。这样第一个易程序就顺利运行完毕。

关闭这个程序可以点击窗口右上角的关闭按钮。

## 1.6 易程序进阶

经过了第一个易程序的编写，可以初步了解一个程序的编写过程，就是构思程序，设计界面，编写代码。下面就继续编写一个稍复杂的易程序，来熟悉易语言的操作。这个程序是在一个编辑框中显示数值，并且每按一次按钮，编辑框中的数值就加 1 位。程序中对组件的部分属性进行了修改。

第一步，新建一个易程序，在窗口中添加一个编辑框组件和按钮组件。如图 1-17 所示。

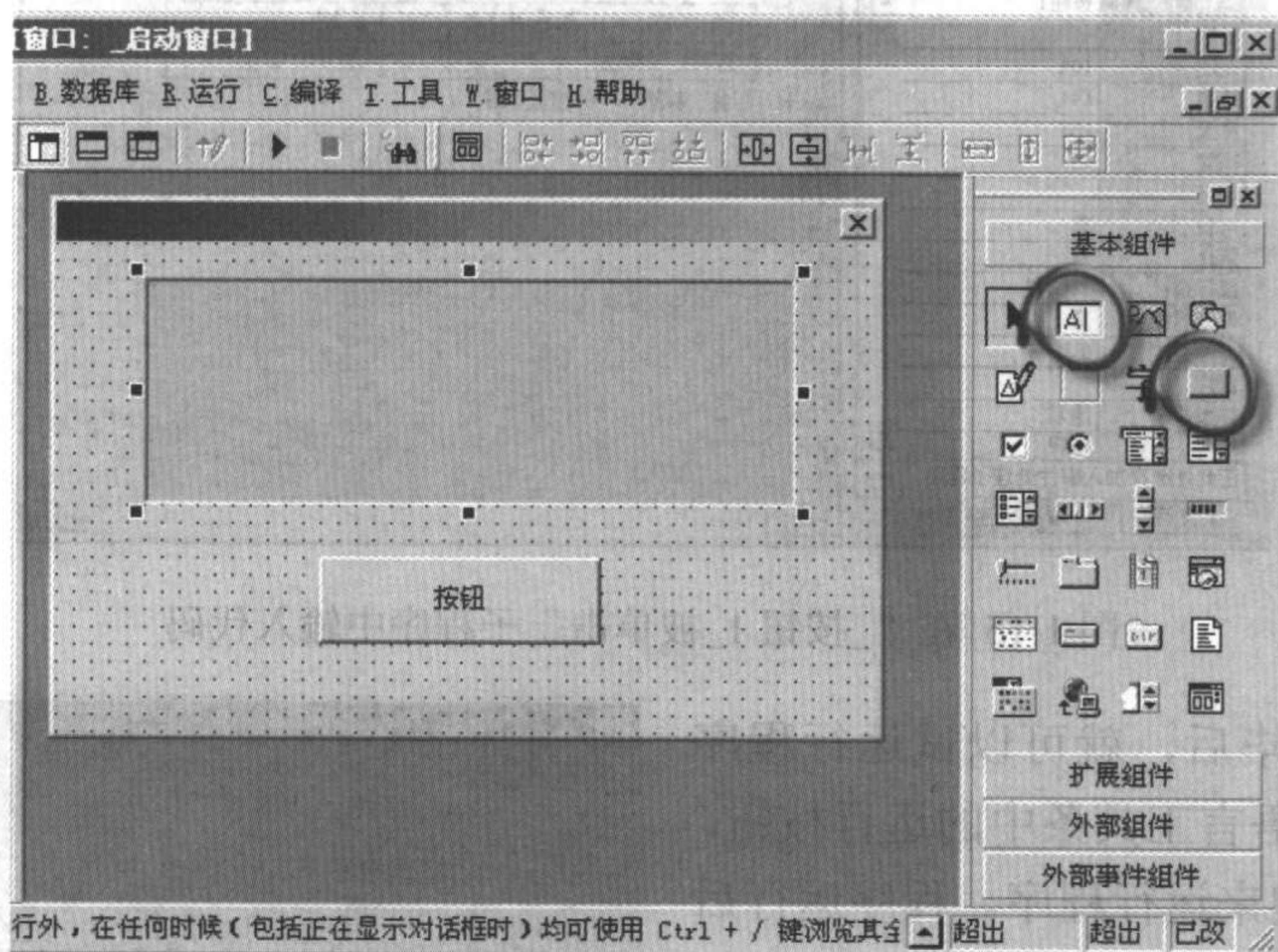


图 1-17 第二个易程序界面

第二步，用鼠标选中编辑框组件，在属性面板中改变编辑框的部分属性：将“输入方式”属性改为“整数文本输入”；改变“文本颜色”、“背景颜色”和“字体”属性。如图 1-18 所示。



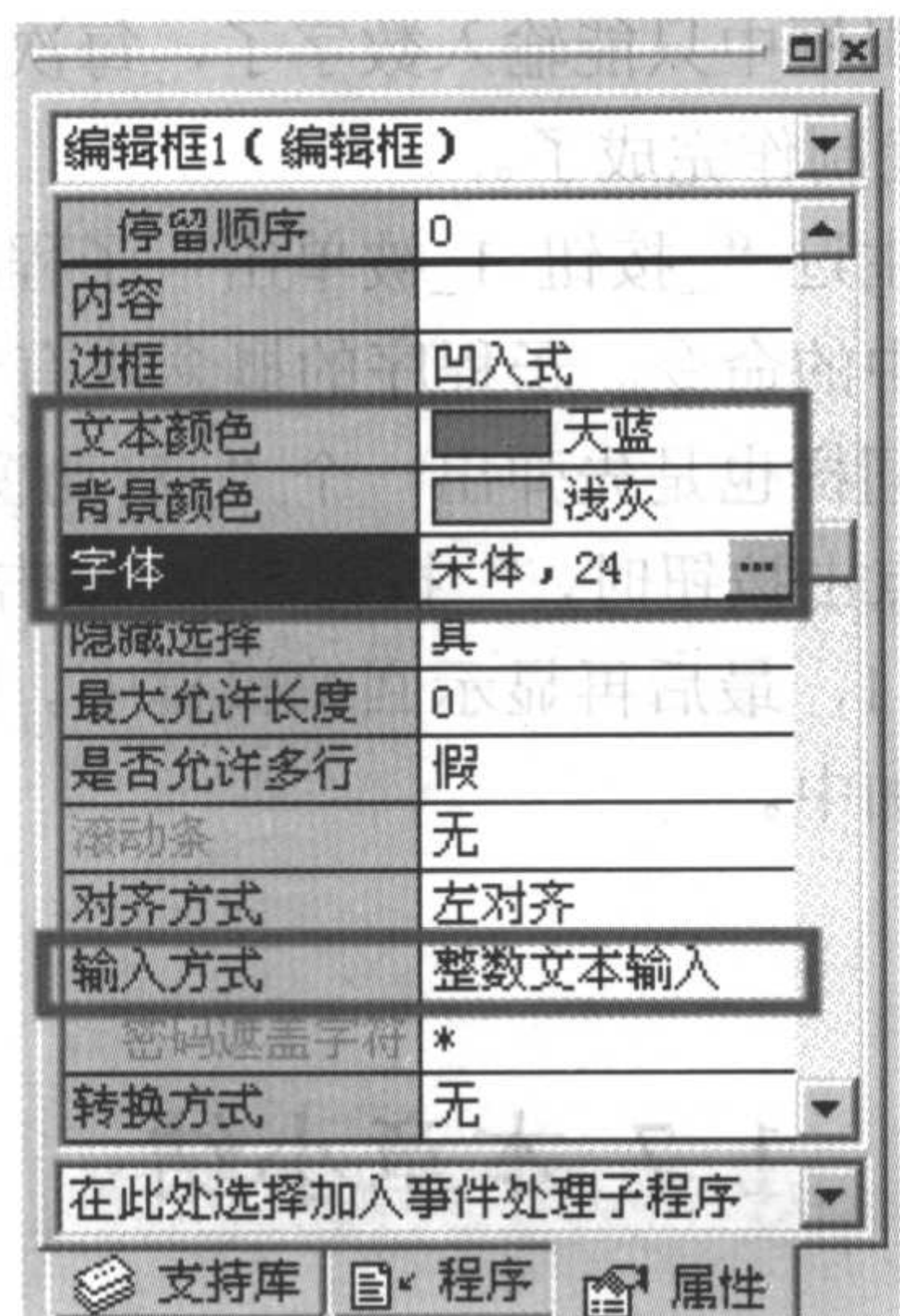


图 1-18 编辑框属性被改变

第三步，双击按钮组件，在“\_按钮 1\_被单击”子程序中输入代码：

编辑框 1.内容 = 到文本 (到数值 (编辑框 1.内容) + 1)

如图 1-19 所示。

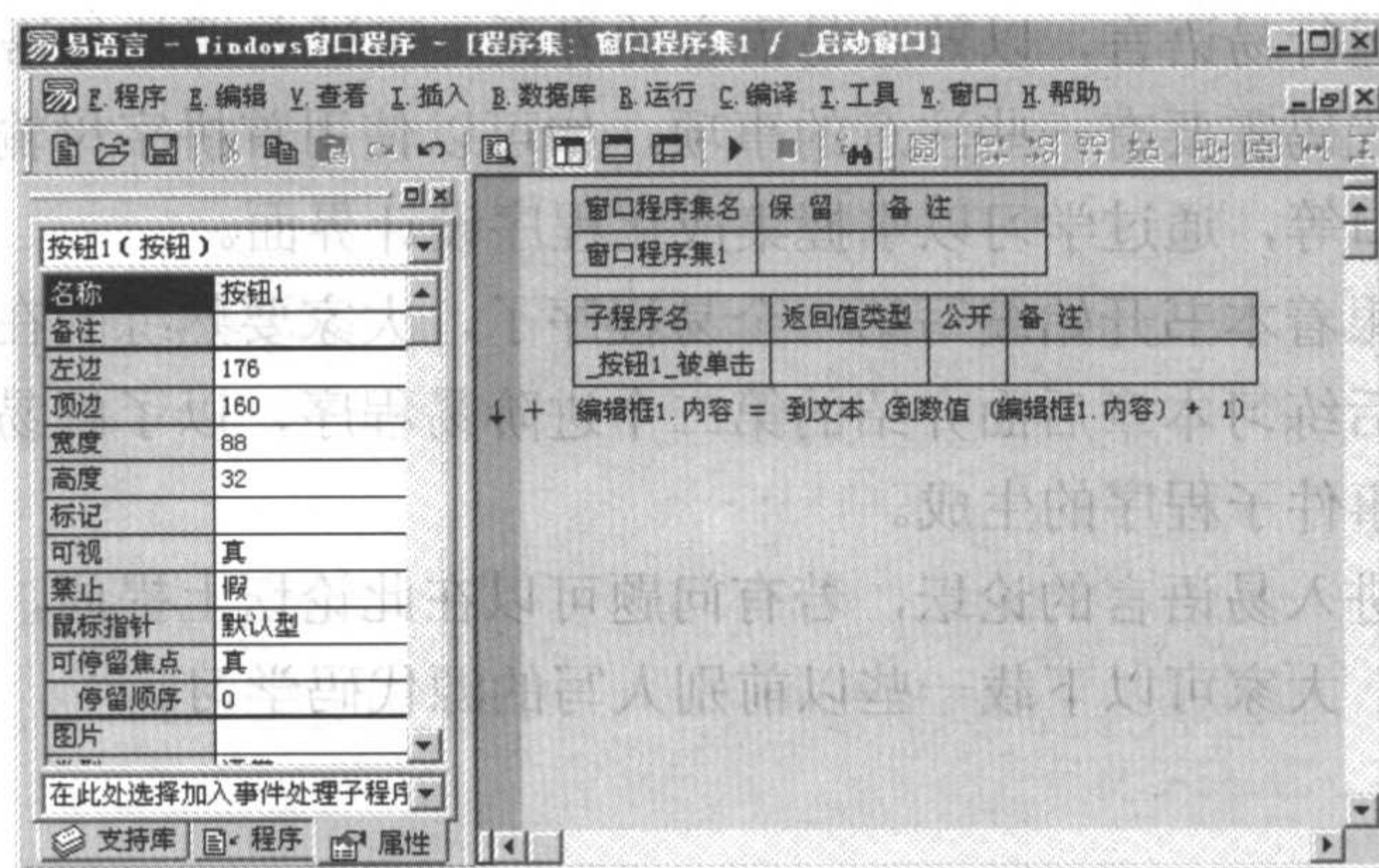


图 1-19 在“\_按钮 1\_被单击”子程序中输入代码

第四步，代码输入完毕后，按下 F5 键试运行程序。如图 1-20 所示。

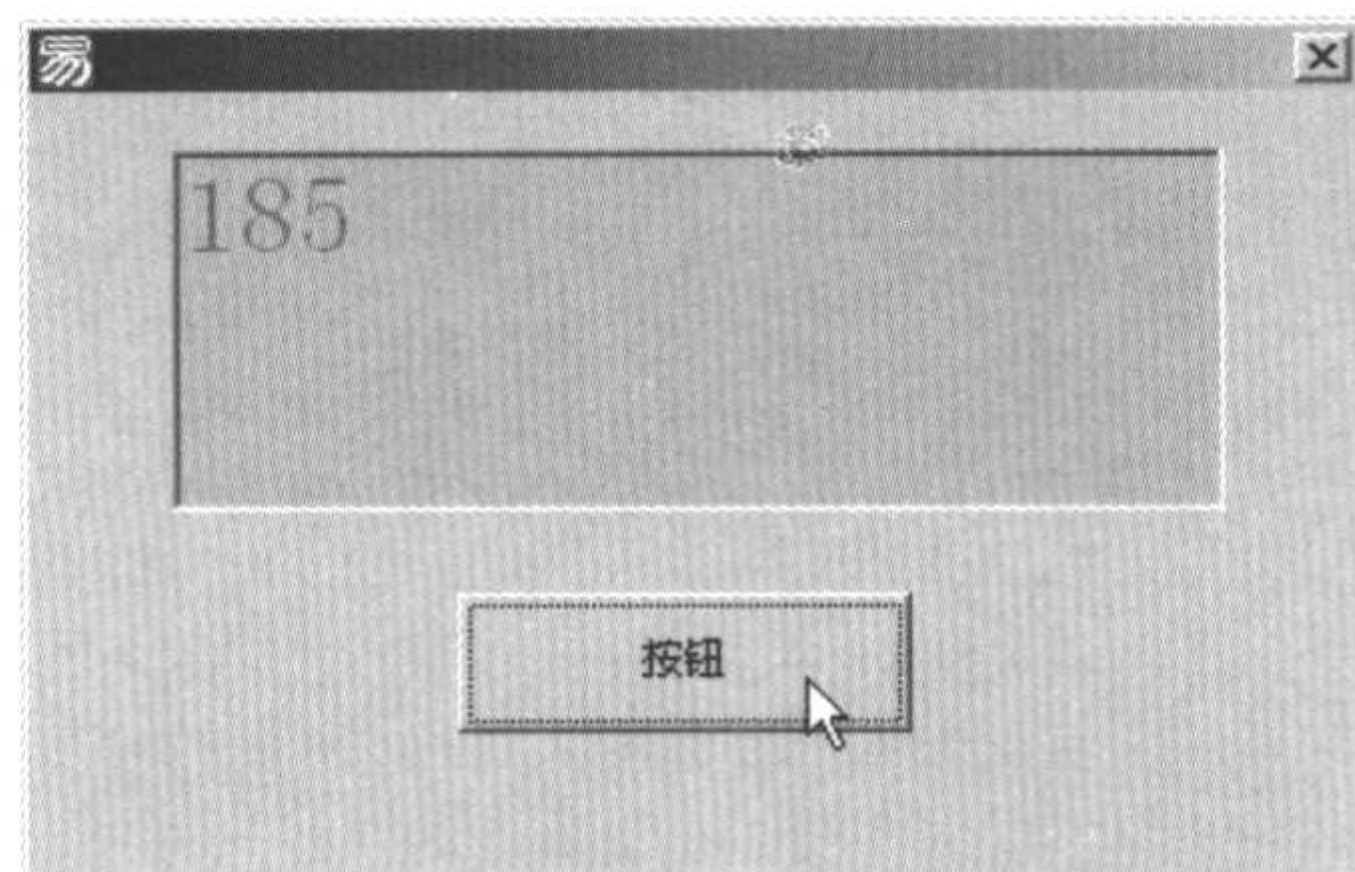


图 1-20 程序运行效果





程序运行后，可以看到编辑框中只能输入数字了，每次点击按钮，编辑框中的数字都会递增 1，一个简单的易程序就制作完成了。

在本书的两个例程中都产生过“\_按钮 1\_被单击”子程序，这个子程序会响应按钮被鼠标单击的事件，从而运行其中的命令。子程序的概念将有专门章节讲解。

本程序的执行过程如下：同样也是先弹出一个“\_启动窗口”，这个窗口中有编辑框组件与按钮组件，当用户用鼠标点击按钮时，“\_按钮 1\_被单击”子程序中的代码被执行。代码将取得编辑框中的内容并加 1，最后再显示回编辑框中。因此大家看到的就是编辑框中的数字通过点击按钮在一直增加中。

### 1.7 本章小结

易语言与其他编程语言基本上一样，但为中国用户定制了一些特色的功能，在此需要简单了解易语言的功能特色。如：易语言内置的输入法就是解决了中文输入慢的问题。

易语言的安装过程是较为简单的，只需跟着步骤一步一步安装即可。大家可以自己试着下载、安装易语言最新版。

安装后可立即运行易语言，以熟悉易语言的界面，可试着调整易语言的一些配置。

书写易语言的代码需要有一些注意的事项，如可以使用首拼字母输入，可以展开参数输入，如何切换窗口等，通过学习以掌握集成化程序设计界面。

接下来就可以跟着本书开始编写第一个易程序了，大家要熟练的在窗口中添加组件和更改组件属性。最后练习本章后面介绍的第二个进阶易程序，以了解易语言其他组件的添加，属性的修改，事件子程序的生成。

课后，大家可进入易语言的论坛，若有问题可以在此论坛上提问，主页中还有一个易语言资源网的连接，大家可以下载一些以前别人写的源代码学习。



## 第二章 数据类型、运算符与表达式

各种数据存放在磁盘或内存中都有其不同的存放格式，因此就存在不同的数据类型。了解各种数据类型的特性，对编程开发来说是十分重要的。

程序中经常会进行一些运算，易语言中的运算都要使用运算符进行识别处理，并通过运算表达式来完成运算操作。程序中对各数据之间的关系的描述也要通过运算符。

本章将对数据类型、运算符和表达式这些程序中基本的内容进行介绍。

### 2.1 易语言的数据类型

一个程序内部应包括两个方面的内容：1. 数据的描述、2. 操作步骤，即对程序动作的描述。

数据是程序操作的对象，操作的结果会改变数据的内容。打个比方：要做一道菜，做菜前先选择烹饪的原材料（即对数据进行描述），然后开始烹饪（即对数据进行操作），最后做好了一道菜（改变了原先数据的状况，得出计算结果）。

编程也一样，程序要对一些数据进行操作，在操作前要先对被操作数据进行描述，即定义相关数据类型的变量，然后再用命令或者方法来对该数据进行操作，最后得到操作结果，进一步可将结果显示出来。

易语言的数据类型从数据结构来区分，可分为基本数据类型和复合数据类型。基本数据类型包括：数值型、逻辑型、日期时间型等；复合数据类型包括所有用户和支持库自定义数据类型。

数据类型可以用来描述变量的类型或组件属性的类型等等，对于变量及常量将在下一章进行讲解，本章主要介绍易语言提供的各种数据类型。

### 2.2 基本数据类型

#### 2.2.1 了解基本数据类型

易语言中基本数据类型有 6 种，包括数值型、逻辑型、日期时间型、文本型、字节集型、子程序指针型。

数值型数据又包括了字节型、短整数型、整数型、长整数型、小数型、双精度小数型。





这些类型代表的数值范围，及机内表示的长度，可以查看“数据类型的长度与溢出”小节中的“表 2-1 数据类型的长度和占用字节”。

**字节型。**可容纳 0 到 255 之间的数值。

**数值型中整数型数据**，如：13556。

**逻辑型数据**，只能有 2 种值，即“真”或“假”。

**日期时间型数据**，用来记录日期及时间，如：[2002-2-2]。

**文本型数据**，可用来记录一段文本，如：“中文编程易语言”。在程序中表示一段文本数据，都要用双引号将文本引起来。

**字节集型数据**，用作记录一段字节型数据，表示为{2,3,4,5}。图片或 mp3 格式的文件是典型的字节集型数据，在程序中，存放此类数据的变量一定要定义为字节集型。

**子程序指针型数据**，是一个子程序在内存中的地址。

## 2.2.2 给变量和返回值定义数据类型

### 1. 给变量定义数据类型

给变量定义数据类型，只需要在新建的变量类型上按下空格键，就会弹出数据类型下拉列表，在列表中选择欲定义的数据类型即可（如图 2-1 所示），也可用拼音或英文在其上直接输入，例如：整数型即输入 zsx 或用英文输入 int。

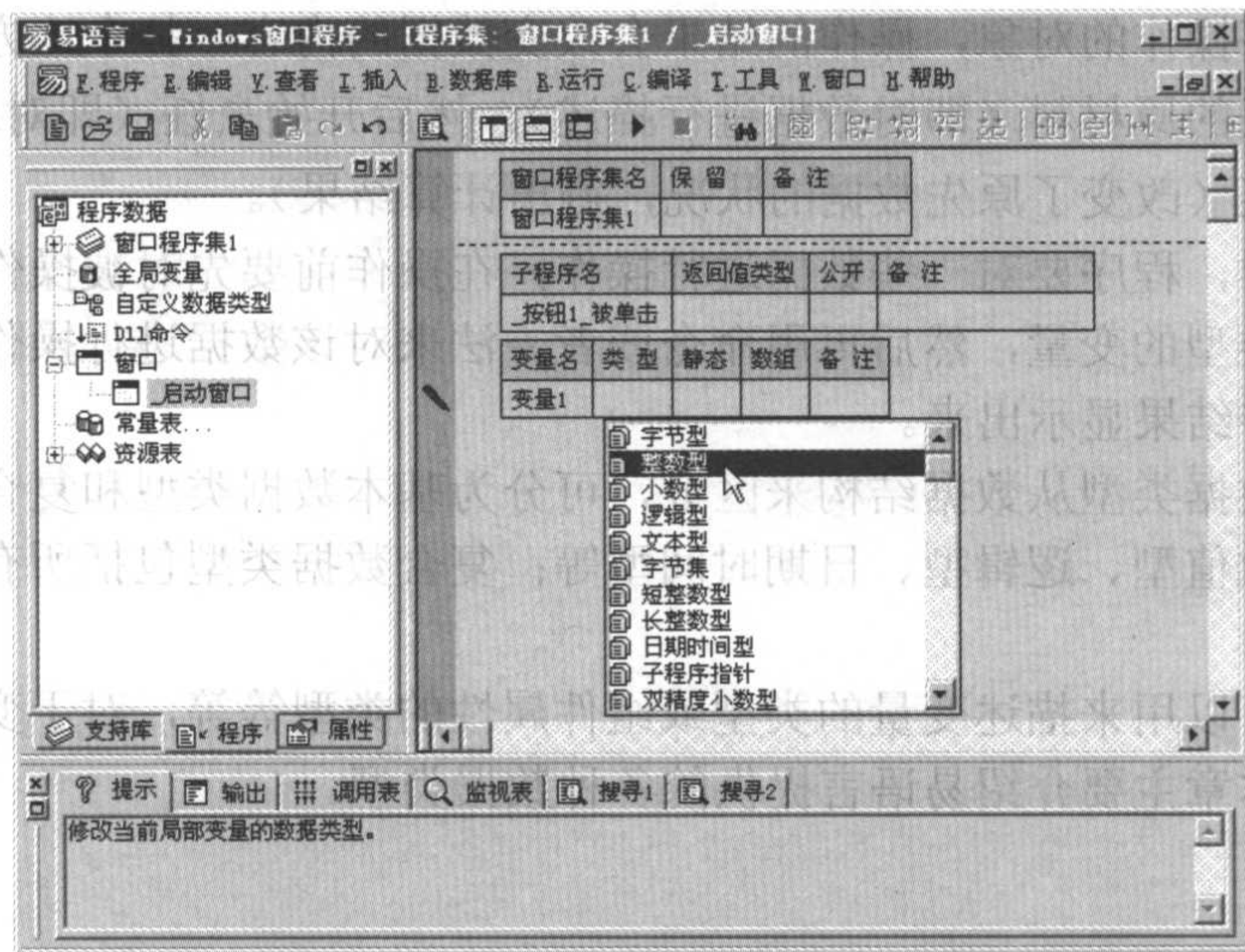


图 2-1 给变量定义数据类型

### 2. 给子程序返回值定义数据类型

和给变量定义数据相同，只需要在子程序的返回值类型上用定义变量的方法定义即可。给子程序定义了返回值类型，就表示该子程序具有了返回值，所以在子程序中编写代码时要记得给子程序返回一个值。

如果易语言新版本增加了新的数据类型，也可在数据类型的下拉列表中查询增加了哪



种数据类型。

### 2.2.3 给数据正确赋值

#### 1. 给组件属性赋值

每个组件都有其属性，每个属性都有自己固定的数据类型，给属性赋值的时候就要考虑到属性的数据类型，比如：标题、名称、内容属性都是文本型的；高度、宽度属性都是整数型的；可视、禁止属性都是逻辑型的。

如果在组件的属性窗口给属性赋值，很多属性都是弹出一个下拉菜单，然后在菜单中选择该属性提供的选项，所以一般不会出错；但使用代码给属性赋予值的时候，就要注意所赋值一定要和被赋值属性的数据类型相符。比如，使用代码给文本型属性赋值时，就要用双引号将欲赋的值引起来：

编辑框 1. 内容 = “中文编程易语言”

如果给逻辑型属性赋值的时候，只能有 2 种值，“真”或“假”：

按钮 1. 禁止 = 真

给整数型属性赋值的时候，直接赋予要改变的值即可：

编辑框 1. 高度 = 100

还有一些需要注意的属性，例如：

(1) 和颜色有关的属性，像“背景颜色”、“文本颜色”等，都是整数型，程序中给此类属性赋值，都是赋予一个颜色值。常用的颜色值在易语言中都作为常量提供，可以直接用“#”+要赋值的颜色名即可。颜色值也可以使用“取颜色值( )”命令来获得，例如：

标签 1. 背景颜色 = # 蓝色

标签 1. 背景颜色 = 取颜色值 (122, 90, 23)

(2) 和图片有关的属性，像“底图”、“图片”等，这类属性一般都是字节集型的，在程序中，直接给这类属性一个图片即可。例如：

图片框 1. 图片 = # 图片 1

如果要将图片框中的图片清除，可以直接给图片属性一个空字节集，空字节集用一对大括号表示，例如：

图片框 1. 图片 = { }

(3) 给日期时间型的属性赋值，月历组件和日期框组件中有很多日期时间型的组件，给这类属性赋值，要输入“[一年一月一日]”，方括号中间是要赋值的时间，例如：

月历 1. 今天 = [2004 年 1 月 15 日]

或输入：

月历 1. 今天 = [2004-1-15]

(4) 整数类型的属性，在属性面板中，该类属性都有多个选项，并且每个选项前都有一个整数，像编辑框的输入方式属性。如图 2-2 所示。



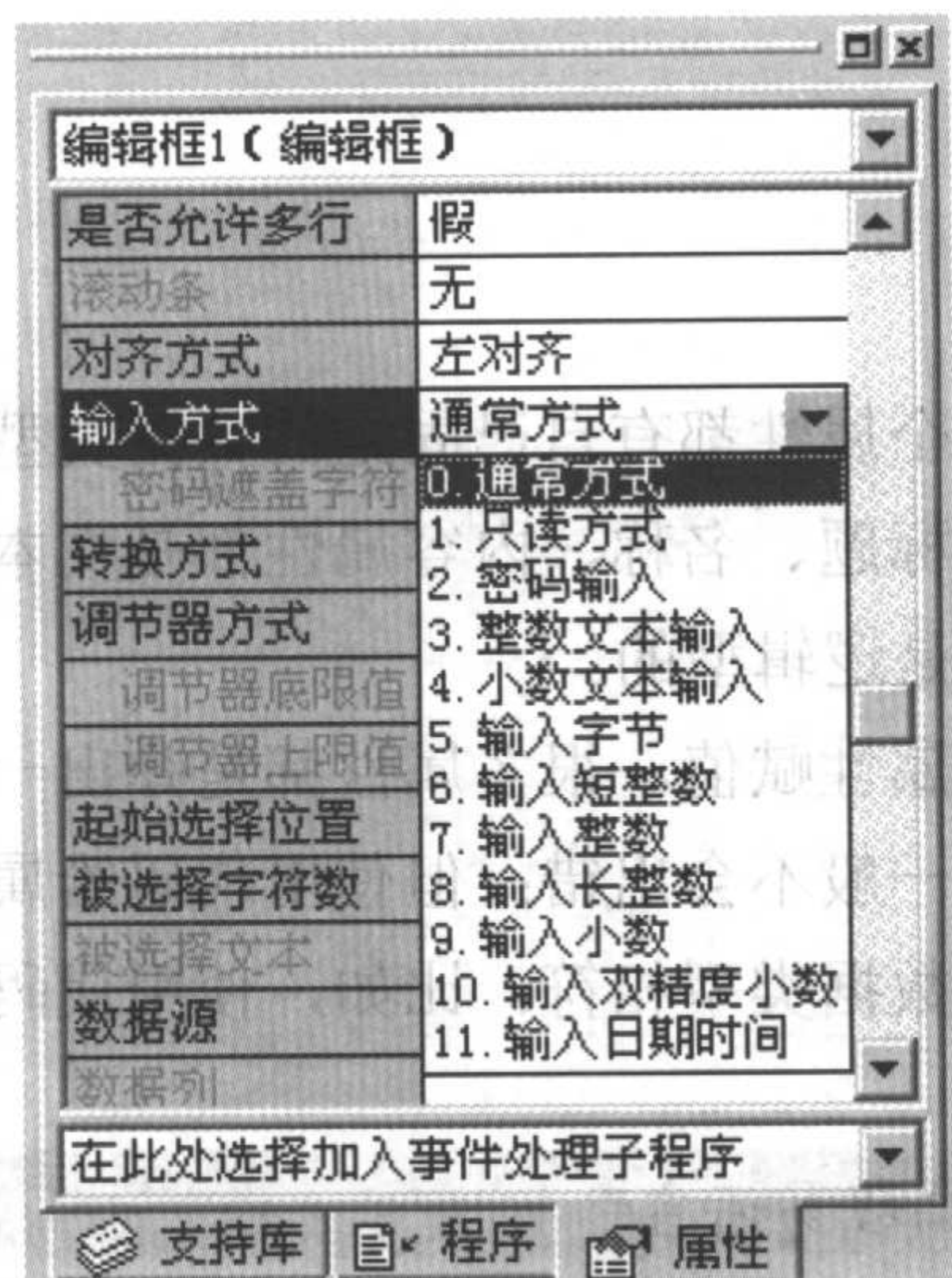


图 2-2 编辑框的输入方式属性

如果在程序中给这类属性赋值，只要输入该属性各选项前的数即可，例如给编辑框的“输入方式”属性改为“密码输入”，需要输入代码：

编辑框 1. 输入方式 = 2

### 2. 给变量正确赋值

和给组件属性赋值的方法相同，定义了变量的数据类型后，要给变量赋值就要注意赋值的类型要和变量类型相同。例如：

变量 1 = “中文编程易语言”      / 给文本变量赋值

变量 1 = 32342      / 给整数变量赋值

变量 1 = [1982 年 1 月 1 日]      / 给日期时间型的变量赋值

变量 1 = # 图片 1      / 给字节集变量赋值, 图片 1 图片资源表中的资源

这里要注意，给“子程序指针”类型的变量赋值，表示为“&”+要指向的子程序名。例如：

变量 1 = &子程序 1

### 3. 计算后赋值

还可以将一段算式的计算结果赋值给变量或属性。例如：

变量 1 = 2 × 3 + 5

编辑框 1. 内容 = 编辑框 2. 内容 + 编辑框 3. 内容

## 2.2.4 数据的比较

在编程中，经常会在各种数据间进行比较。同种数据类型之间进行比较，可以直接进行；而不同种数据之间进行比较，就要先进行数据类型的转换，将不同种的数据类型转换为同一种数据类型后才能进行比较，否则程序会报错。



例如：编辑框中输入了一个整数，要比较编辑框中的内容是否大于 50。由于编辑框中的内容是一个文本，首先要将编辑框中的内容转换成整数型数据后，再进行比较，输入以下代码：

```
到数值 (编辑框 1. 内容) > 50
```

比较后，会返回一个逻辑值，如果大于 50 就会返回真，小于或等于 50 将返回假。

易语言中常用的数据类型间转换的命令有：

“到数值 ()”命令，用来将一个通用型数据转换到整数型。

“到文本 ()”命令，用来将一个通用型数据转换到文本型。

“到字节集 ()”命令，用来将一个通用型数据转换到字节集型。

“从字节集转换 ()”命令，用来将一个字节集型的数据转换成通用型数据，命令的第 2 个参数控制欲转换成的数据类型。

“到时间 ()”命令，用来将一个文本型的数据转换成日期时间型。

使用这些数据类型间互相转换的命令，就可以进行不同数据类型间的比较了。

例如：比较 2 个编辑框中数的大小，用信息框显示出比较的结果，并用第 3 个编辑框显示出较大数减较小数的结果。在窗口中添加 3 个编辑框组件和一个按钮组件，双击按钮组件，然后输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```
如果 (到数值 (编辑框1.内容) > 到数值 (编辑框2.内容))
    信息框 ("编辑框1中数大,数值为:" + 到文本 (编辑框1.内容), 0, )
    编辑框3.内容 = 到文本 (到数值 (编辑框1.内容) - 到数值 (编辑框2.内容))
    如果 (到数值 (编辑框1.内容) = 到数值 (编辑框2.内容))
        信息框 ("2个编辑框中的数相等,数值为:" + 到文本 (编辑框1.内容), 0, )
        信息框 ("编辑框2中的数大,数值为:" + 到文本 (编辑框2.内容), 0, )
        编辑框3.内容 = 到文本 (到数值 (编辑框2.内容) - 到数值 (编辑框1.内容))
```

变量之间的比较也是一样，一定要注意变量的数据类型，不同数据类型的变量一定要转换成相同类型后再进行比较。

## 2.2.5 数据类型的存储字节与溢出

### 1. 数据类型的存储字节

各种数值型的数据都在内存中占用一定的存储空间。字节(byte)是系统中的基本存储单位。数据类型所占字节数越多，所能够容纳数值的范围就越大。参见表 2-1。

表 2-1 常用数据类型

数据类型名称	占用字节数	取值范围
字节型	1	0 到 255
短整数型	2	-32,768 到 32,767





数据类型名称	占用字节数	取值范围
整数型	4	-2,147,483,648 到 2,147,483,647
长整数型	8	-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807
小数型	4	-3.4E38 到 3.4E38 (7 位小数)
双精度小数型	8	-1.7E308 到 1.7E308 (15 位小数)
逻辑型	2	“真”或“假”
日期时间型	8	100 年 1 月 1 日到 9999 年 12 月 31 日
子程序指针	4	尺寸为 4 个字节。此数据类型的数据用来与外部程序或操作系统 API 进行交互，是一个子程序在内存中的地址。
文本型		由以字节 0 结束的一系列字符组成
字节集		一段字节型数据

从上表可以看出，数值型的数据容纳的数值范围越大，占用的字节就越多。比如，短整数型的数值 3000 和整数型的数值 3000，都代表了数值 3000，但在系统中占用的空间却不同，即短整数型占 2 个字节，整数型占 4 个字节。所以，在实际应用时就要根据自己的需要来选择使用的数据类型，避免存储空间的浪费。例如，存储的数据在-32768 至 32767 以内，就要采用短整数型；如果使用小数而对精度不高，就可以使用小数型而不用采用双精度小数型等等。

## 2. 数据的溢出

某数据类型存储的值超出了其所能容纳的范围，就会发生数据溢出错误。比如，让短整数型数据存放大于 32767 的数值，将会得到错误的结果。所以在选择数据类型时，除了要避免空间的浪费，又要防止数据的溢出。

可以做一个简单例程来测试一下数据的溢出，新建一个易程序，然后在窗口中添加 2 个编辑框组件和 1 个按钮组件，然后用鼠标双击按钮组件，在“\_按钮 1\_被单击”的子程序中首先按下 Ctrl+L 键，新建一个变量，并定义变量名为“变量 1”，变量类型为整数型，然后输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
变量1	整数型			

变量1 = 到数值 (编辑框1.内容)

编辑框2.内容 = 到文本 (变量1)

最后试运行程序，在编辑框中输入一个很大的数（比如超过十位），然后点击按钮，如果超出整数型变量容纳范围而产生溢出，编辑框会显示出错误的结果。



## 2.3 特殊数据类型

### 2.3.1 了解特殊数据类型

易语言中提供的每一种内部组件都可以作为一种数据类型，比如：图片框、编辑框、按钮等等。这些数据类型具有组件的特征，如属性、方法等等都和组件完全相同，在程序中也可以当成一个组件来使用。例如：窗口中有 1 个画板组件“画板 1”，程序中可以直接将该画板组件赋值给一个数据类型为“画板”的变量 1，使用代码：

```
变量 1 = 画板 1
```

这样，变量 1 就具有了画板的属性、方法。在程序中表示该变量的属性方法如下：

```
变量 1. 高度 = 100
```

```
变量 1. 宽度 = 200
```

### 2.3.2 动态添加组件

利用这些特殊的数据类型可以实现动态的向窗口中添加组件，使用“复制窗口组件( )”命令。被复制的组件必须是窗口中已经存在组件。例如：程序中添加了 1 个画板组件和一个按钮组件，当按钮被按下后就在窗口中再添加一个画板组件，然后改变动态添加的画板组件的属性，该程序所用到的代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
变量1	画板			

```
复制窗口组件 (画板1, 变量1)
```

```
变量1.顶边 = 0
```

```
变量1.左边 = 0
```

```
变量1.画板背景色 = #蓝色
```

```
变量1.可视 = 真
```

## 2.4 自定义数据类型

除了使用易语言提供的数据类型以外，还可以根据需要自定义新的数据类型。例如要定义一个数据类型“矩形”，定义方法如下：

第一步，新建一个易程序，在程序面板双击“自定义数据类型”。如图 2-3 所示。程序会自动切换到“自定义数据类型”的界面。



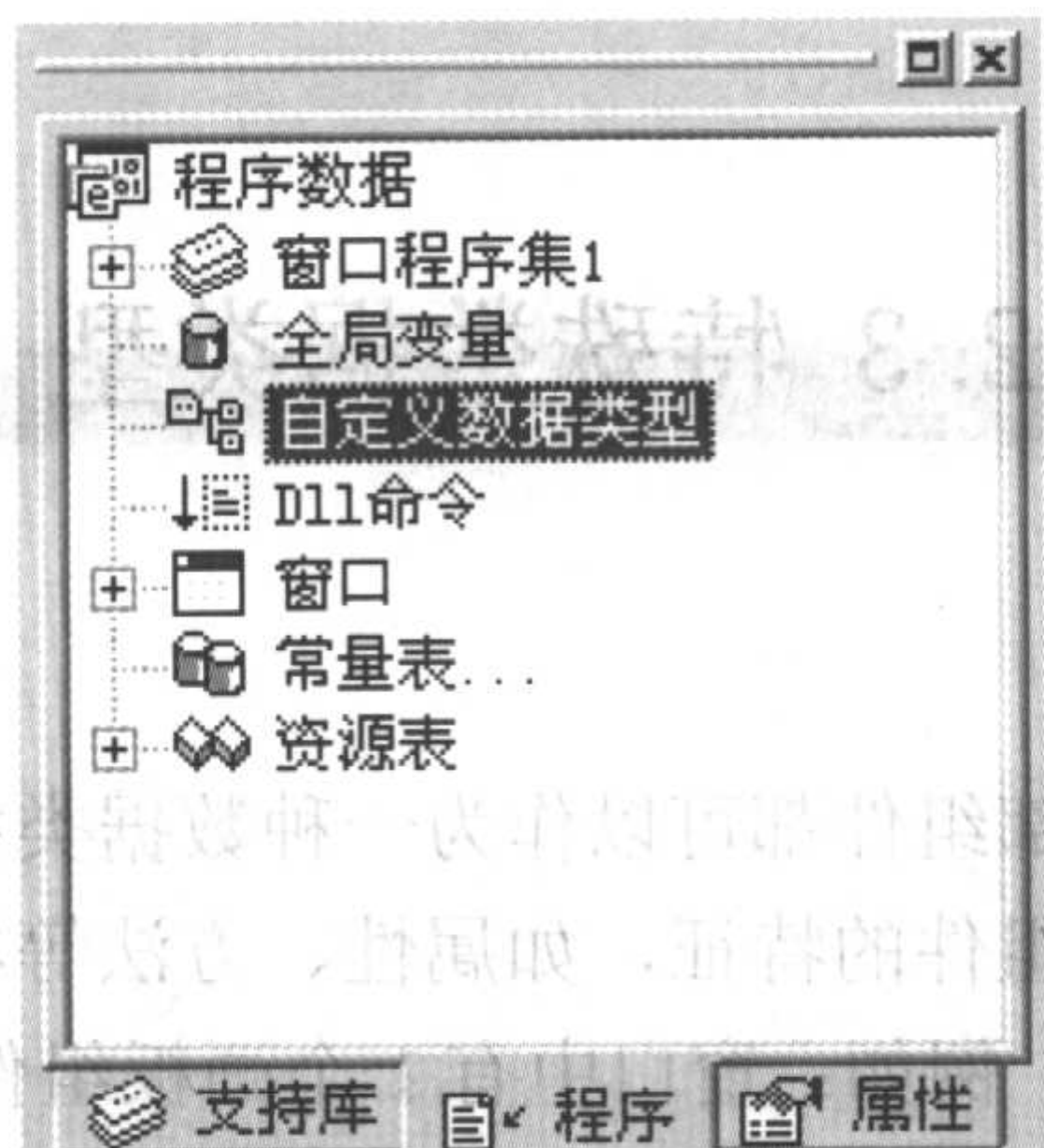


图 2-3 自定义数据类型

第二步，在自定义数据类型界面按下 Ctrl+N 键，新建一个数据类型。然后将数据类型名定义为“矩形”，由于决定一个矩形的位置取决于矩形左上点的横纵坐标和矩形右下点的横纵坐标。所以，在“成员名”上按 4 次回车，加入 4 个成员。将 4 个成员名分别定义为“左上横坐标”、“左上纵坐标”、“右下横坐标”、“右下纵坐标”。如图 2-4 所示。

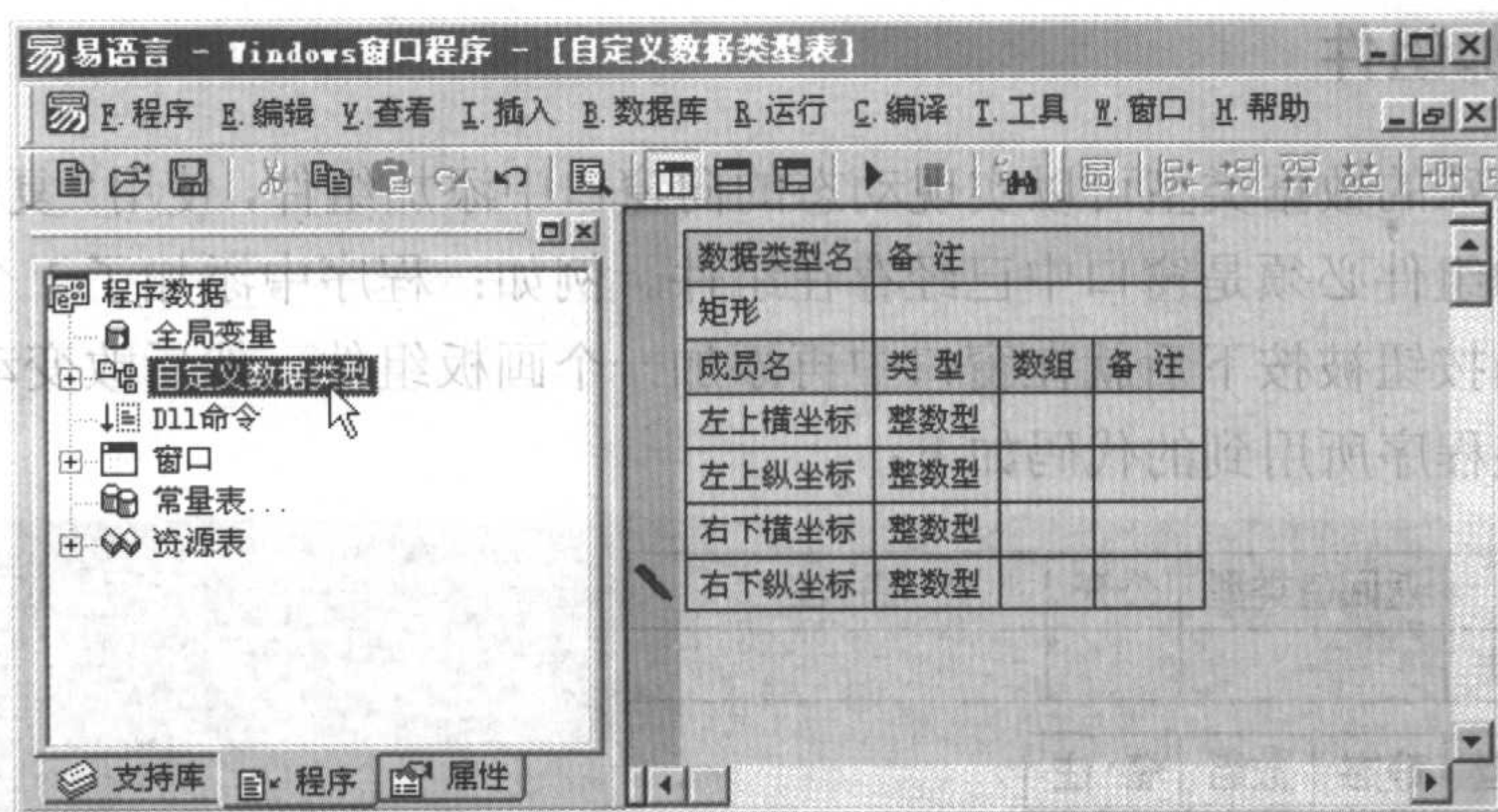


图 2-4 定义“矩形”数据类型

最后，就要来使用这个自定义的数据类型了。画板有一个方法是“画矩形()”，下面就用自定义的数据类型“矩形”来为画矩形方法填写参数。在窗口中添加一个画板组件和一个按钮组件，双击按钮组件，在“\_按钮 1\_被单击”子程序中新建一个变量，变量名为“矩形”，然后定义变量的数据类型为“矩形”，然后输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
矩形	矩形			

矩形.左上横坐标 = 画板1.顶边

矩形.左上纵坐标 = 画板1.左边

矩形.右下横坐标 = 画板1.宽度

矩形.右下纵坐标 = 画板1.高度

画板1.画矩形 (矩形.左上横坐标, 矩形.左上纵坐标, 矩形.右下横坐标, 矩形.右下纵坐标)



代码输入后试运行程序，点击按钮，画板会以画板的边框大小画一个矩形。完整的程序请参见随书光盘中的例程“画矩形.e”

## 2.5 易语言运算符

编写代码时，除了大量的使用命令或对组件的属性或方法进行操作，运算符的使用也非常重要。程序中所有涉及的算术运算或关系比较运算等操作，都需要使用运算符。

易语言中提供了大量的运算符。例如赋值时使用的“=”号，就是赋值运算符，比较大小时使用的“>”和“<”号，是关系运算符等等。如表 2-2 所示。


表 2-2 易语言运算符

运算符分类	运算符	运算符含义	代码中显示
算术运算符	+	加法运算，将加号两边的数相加	+
	-	减法运算，将减号两边的数相减；负号	-
	*	乘法运算，将乘号两边的数相乘	×
	/	除法运算，将除号两边的数相除	÷
	\	整除运算，将整除号两边的数整除	\
	%	求余数运算	%
关系运算符	>	判断是否大于	>
	<	判断是否小于	<
	= 或 ==	判断是否等于	=
	>=	判断是否大于等于	≥
	<=	判断是否小于等于	≤
	<> 或 !=	判断是否不等于	≠
	?=	判断是否约等于	≈
逻辑运算符	&& 或 qie	逻辑与运算符，可以连接几个必须同时	且
	或 huo	逻辑或运算符，可以连接几个可选条件	或
赋值运算符	=	将等号后面的值赋值给等号前面的对象	=

程序中的运算符都有其优先级别，在程序运行的时候会按照符号的优先级别，从高到低依次运行。运算符的优先级别参见表 2-3 易语言常用运算符的优先级。



表 2-3 易语言常用运算符的优先级

运算符	优先级
() (小括号)	最高
*(乘) / (除)	
\ (整除)	
%(求余数)	
+(加) -(减)	
< (小于) <= (小于等于) > (大于) >= (大于等于)	
==(等于) != (不等于) ?= (约等于)	
&& (逻辑与)	
(逻辑或)	
= (赋值)	

本章将只介绍运算符和赋值运算符，其他的运算符将在以后各章中结合有关内容陆续介绍。

## 2.6 运算符与表达式

### 2.6.1 算术运算符和算术表达式

#### 1. 算术运算符

算术运算符上节表中已经介绍了。在程序中表示为：

＋，加法运算。如：3+2

－，减法运算或负值运算。如：10-2、-10

×，乘法运算。如：2×3

/，除法运算。如：20/12

\，整除运算。如：12\5，运算后会将保留一个整数，小数部分将被舍去

％，余数运算。还可以输入“求余数”，第一个参数填被除数，第二个参数填除数，第二个参数可以重复添加。如：1220%100、1220%100%120

#### 2. 易语言算术表达式

用算术符号和括号将运算对象连接起来的，符合易语言语法规则的式子，称易语言算术表达式。例如，下面是一个合法的易语言算术表达式：

变量 1 = ((6 × 12 + 16 ÷ 8) - 23) \ 10

表达式中运算的先后，是按照运算符的优先级别来进行判定的。



算式计算的结果可以被程序调用，例如：将上面算式的结果用信息框显示出来：  
信息框  $((6 \times 12 + 16 \div 8) - 23) \setminus 10, 0, )$

### 2.6.2 赋值运算符和赋值表达式

1. “=”是赋值运算符，在程序中给变量赋值或用代码改变组件属性，大部分都是使用“=”进行赋值的，将等号后面的值赋值给等号前面的赋值对象。例如：

变量1 = 100

编辑框.高度 = 200

2. 赋值表达式。一个正确的赋值表达式，一定要保证欲赋的值和被赋值的对象之间的数据类型相同，不同的数据类型要转换成相同的数据类型后再赋值。

3. 赋值运算符“=”和关系运算符“=”的区别。虽然2个运算符使用的是相同的符号，但含义却不同，赋值运算符“=”是用于赋值，将“=”右边的值（或变量）赋值给“=”左边的变量（或组件属性、数组成员、自定义数据类型成员）；关系运算符“=”，是比较符号两边的值是否相等，如果相等返回真，不相等返回假。例如：

变量名	类型	静态	数组	备注
变量1	整型			
变量2	整型			

```

-- 如果 (变量1 = 变量2)
-- 标签1.标题 = “相等”
-- 标签1.标题 = “不相等”

```

上述代码中，条件语句“如果”中的“变量1 = 变量2”，是用关系运算符“=”进行比较，如果相等会返回“真”，不相等会返回“假”，如果返回“真”将会执行：标签1.标题 = “相等”；如果返回“假”将会执行：标签1.标题 = “不相等”，这2行给标签标题属性赋值的代码中，使用的就是赋值运算符“=”。

注：赋值表达式的形式固定为单一类似“xxx = xxx”的语句，其他任何形式下的“=”运算符都为比较运算符。

## 2.7 本章小结

虽然大家有可能对数据类型不能一下子了解透彻，但需要将常用的数据类型记住，等后面的章节会讲得更多，大家就可以作进一步的了解了。

本章中的某些例程较为简单。如对数据类型的溢出测试只测试了一项，大家可以自由改程序中的数据类型，将内容赋值超过数据的范围，从而得到溢出后的测试结果。大家也可以新建变量而设置完数据类型后不赋值，并用信息框显示出来，这样可以查看不同数据



类型的初始值。

大家还可以进行以下练习：

1. 新建易程序，试着在窗口中添加一些组件，然后分别在属性面板中和程序代码中改变组件的属性，查看属性修改后的效果。
2. 学会新建变量和指定变量的数据类型。
3. 试着给不同类型的变量赋值。
4. 了解和使用易语言的运算符。使用易语言写几行表达式代码来解决常见的数学问题，作四则运算法的计算，注意使用括号将先要计算的内容括起来。
5. 试动态在一个新建易程序中的窗口内添加组件，并显示出来。



## 第三章 变量、常量与资源

程序运行过程中所存放的临时数据主要通过变量保存，编写程序离不开变量的使用，通过本章的学习，应掌握变量的定义方法和使用方法。

常量，顾名思义，它的值是固定不变的。易语言的核心支持库中已经定义了大量常量，其他支持库通常也会定义一些常量，用户也可以在程序中定义自己的常量。

各种图片或声音等资源都被看做常量，要想在程序中随时调用，可以将其存放在易语言的资源表中，这样在编程时不但可以随时调用，而且资源表中的资源会和程序一起编译到可执行文件里面。

本章将介绍变量、常量和易语言资源表的使用方法和应用实例。

### 3.1 变量

#### 3.1.1 了解变量

变量具有一个名字，变量可以存储可变的数据，变量在内存中占据了一定的存储单元。单元中存储的值是该变量的值。

其实，完全可以把变量看做一个容纳东西的容器，只是这个变量容纳的是各种可变的数据。打个比方来说，可以将变量看做是买东西时用的手提包，买完东西后就将东西放到包里，包里的东西当然是可以改变的，可以拿出来，也可以放进去。变量也是一样，可以提取变量中的数据，也可以改变变量中的数据。

上一章中涉及的变量，都将变量名称定义成了“变量1”，这是做简单的定义。其实变量名是可以任意定义的，也可以根据需要，给变量起一个有实际意义的名字，如：“存放鼠标横坐标的变量”或直接定义变量名为“鼠标横坐标”，这样定义变量名，可以在日后查看代码时很快了解该变量的作用，并且方便与他人交流代码。在变量使用很多的时候，定义变量名尤为重要。养成一个良好的定义名称的习惯，对以后的编程将有很大帮助。但定义变量名时也要注意：首字符不可以是数字，并且变量名中除“\_”以外，不可以使用其他的符号和标点，这些都是为了防止和程序中的数值和符号重复，造成程序的混乱。

变量在程序中的应用是非常广泛的。变量不但可以用来存放各种类型的数据，还可以作为命令的参数使用，如用于“计次循环首( )”命令，命令的第二个参数变量会记录循环的次数。





### 3.1.2 变量类型

变量有几种类型，不同种类的变量的特性也有所不同。从变量的作用范围来区分，可以将变量分为“局部变量”、“程序集变量”和“全局变量”。

**局部变量**，只能在其所在的子程序中才能被调用的变量，其他子程序都无法调用。因为子程序被调用的时候，这种变量才占用系统的内存，当子程序执行结束后，变量所占空间被系统收回，因此局部变量是非常节省系统内存的。

**程序集变量**，一般情况下仅在本程序集中被调用。若在其他窗口程序集中调用，则需要在变量名前加程序集所对应的“窗口名称”前缀，例如：

信息框(-启动窗口. 变量1, 0, )

程序集变量所在的程序集中的所有子程序，都可以自由访问程序集变量，多个子程序都需要访问的数据，可以使用程序集变量来存储。

**全局变量**，在程序运行后，所有程序集内子程序都可以使用的变量。也是覆盖范围最大的变量。这种变量在程序运行后即占用内存空间，在程序运行结束才从内存中清除，所以会长时间占用系统资源，建议根据程序的实际情况适当使用。

在选择使用变量的类型时，尽量选择符合该变量使用范围的变量类型，以节省系统内存。

从变量的属性来区分，还可以将变量分为“静态变量”和“数组变量”。

**静态变量**，就是静止存在的局部变量。当所处子程序退出时，静态局部变量能够保留住现行内容以供下次继续使用；而非静态的变量就不能，下次进入子程序时它将被重新初始化。如果局部变量不设置“静态”属性，子程序执行完毕后，将清空该子程序中的所有非静态局部变量；如果局部变量设置了“静态”属性，当子程序执行完毕后也不会被清空，当子程序再次被调用时，静态变量的值仍保持上次被调用时的状态。静态变量的使用，将在后面的静态变量章节详细介绍。

**数组变量**，即可以存放一组数据的变量。数组变量中的每个成员都拥有独立的存储单元，可以单独调用和赋值。其实数组变量可以看做是多个非数组变量组成的。数组变量又分为“单维数组变量”和“多维数组变量”。关于数组变量的相关内容将在数组变量章节详细介绍。

定义变量的方法很多，在上一章中，已经涉及了一些变量的定义，比如用 **Ctrl+L** 键新建“局部变量”，并指定变量的名称和数据类型。定义变量还可以用 **Ctrl+G**，新建一个“全局变量”。还可以在易语言的“插入”菜单中选择“新局部变量”和“新全局变量”来插入局部变量和全局变量。如图 3-1 所示。



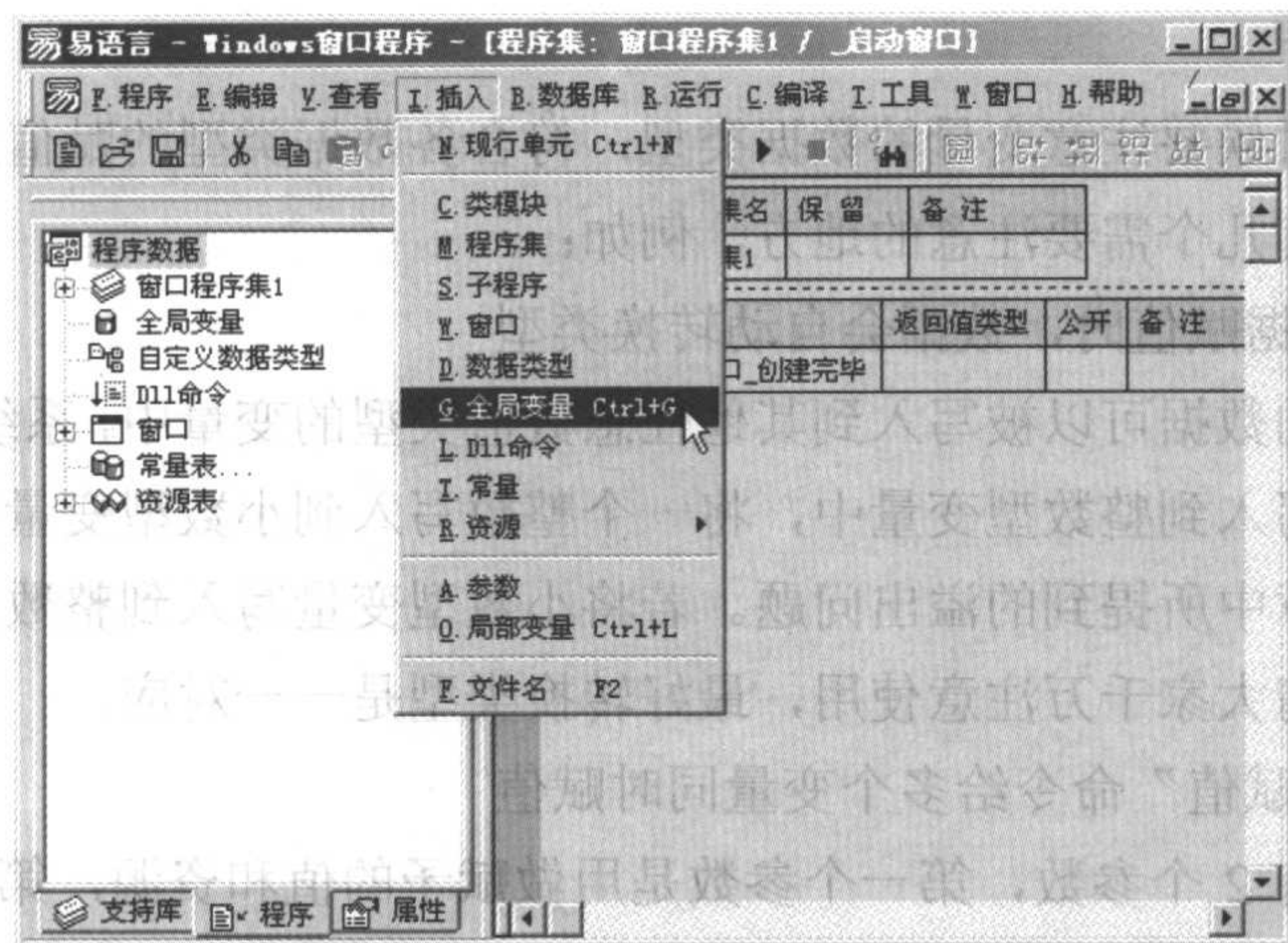


图 3-1 新增全局变量

在“窗口程序集名”上按回车键，可以插入新的“程序集变量”。各种类型的变量将在“变量类型”一节详细介绍。如图 3-2 所示。

窗口程序集名	备注		
窗口程序集1			
变量名	类型	数组	备注
程序集变量			

图 3-2 插入新的程序集变量

如果代码中引用了没有定义的变量，则该行代码不会通过编译，如果按 **Shift+Enter** 键来编译当前行，则状态夹中会有相应的错误提示，大家可以在编写代码时使用一些未定义的变量，但一定要记住在输入代码后再来补建变量，然后将光标移到该行并按 **Shift+Enter** 键来重新编译此行代码。如图 3-3 所示。

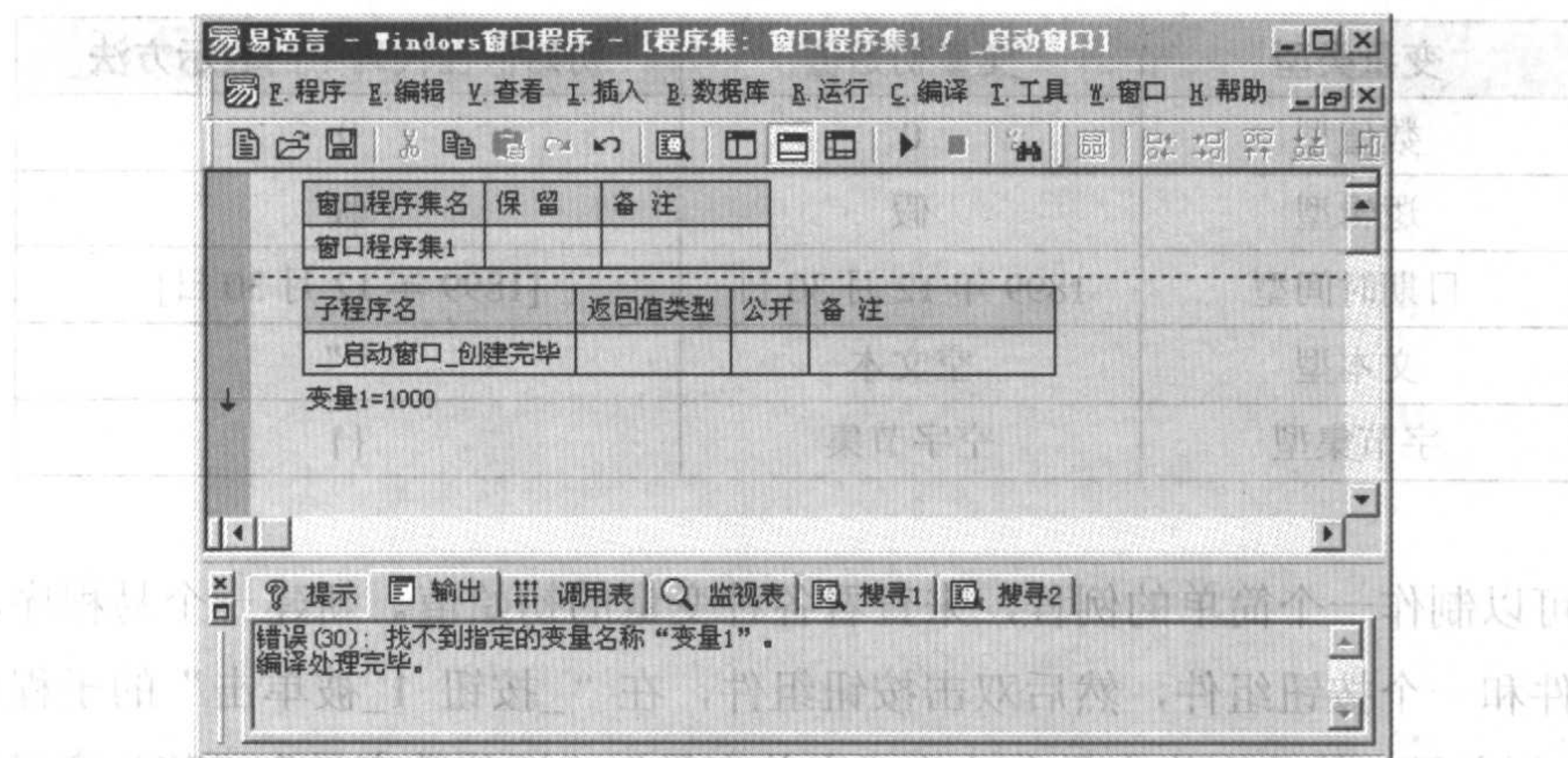


图 3-3 状态条中显示找不到变量



### 3.1.3 变量的赋值

给变量赋值的时候要注意变量的数据类型，符合各数据类型的赋值规则即可。

变量的赋值还有几个需要注意的地方，例如：

#### 1. 给数值型数据赋值时，数据会自动转换类型

任意数值类型的数据可以被写入到其他任意数值类型的变量中，系统将自动进行转换。譬如将一个短整数写入到整数型变量中，将一个整数写入到小数型变量中等等，但是此时必须注意防止上一章中所提到的溢出问题。若将小数型变量写入到整数型变量时，会丢失小数点后的内容，请大家千万注意使用，最好转换类型是一一对应。

#### 2. 使用“连续赋值”命令给多个变量同时赋值

连续赋值命令有 2 个参数，第一个参数是用做赋予的值和资源，第二个参数是被赋值的变量或变量数组，第二个参数可以重复添加，即可以添加多个被赋值的变量，例如：

连续赋值 (100, 变量 1, 变量 2, 变量 3, 变量 4)

命令运行后，将会给变量 1、变量 2、变量 3 和变量 4 同时赋值 100。这一行代码相当于以下四行代码：

```
变量 1 = 100
变量 2 = 100
变量 3 = 100
变量 4 = 100
```

### 3.1.4 变量的初始值

变量的初始值即一种变量在程序运行后，没有赋予新值前的初始数据。每一种数据类型的变量初始值都有所不同。像文本型变量的初始值是一个空文本，表示为“”，数值型变量的初始值是 0 等等。如表 3-1 所示。

表 3-1 变量的初始值

变量类型	变量初始值	初始值在代码中的表示方法
数值型	0	0
逻辑型	假	假
日期时间型	1899 年 12 月 30 日	[1899 年 12 月 30 日]
文本型	空文本	“”
字节集型	空字节集	{}

大家可以制作一个简单的例程，来查看各种变量的初始值。新建一个易程序，添加一个画板组件和一个按钮组件，然后双击按钮组件，在“\_按钮 1\_被单击”的子程序中，新建 10 个局部变量，并分别将变量名改为“字节变量”、“短整数变量”、“整数变量”、“长整数变量”、“小数变量”、“双精度小数变量”、“逻辑变量”、“日期时间变量”、“文本变量”、



“字节集变量”，然后根据变量名定义相应的数据类型，并输入代码：

画板 1. 滚动写行 (字节变量, 短整数变量, 整数变量, 长整数变量, 小数变量, 双精度小数变量, 逻辑变量, 日期时间变量, 文本变量, 字节集变量)

代码输入后试运行程序, 点击按钮, 画板会将 10 个变量的值滚动写在画板上, 由于没有对这些变量进行赋值, 所以画板显示的是这些变量的初始值。参见随书光盘中例程“变量初始值.e”

## 3.2 静态局部变量

“静态”属性是局部变量的重要属性。具有“静态”属性的局部变量称为“静态局部变量”。静态局部变量在子程序运行完毕后仍保留其内容; 而非静态变量, 即普通局部变量, 在每次进入子程序时都被重新初始化。

静态变量大致相当于“局部变量”和“全局变量”的结合: 它具有局部变量的作用域, 同时具有全局变量的生命周期。

静态变量的定义和取消定义的方法很简单, 在欲定义的局部变量的静态属性上点击鼠标左键, 当静态属性上出现“√”后, 即表示定义了一个静态变量, 当再次点击将“√”去掉, 即表示取消定义。也可以在静态属性上按空格键, 也可以定义和取消变量的静态属性。如图 3-4 所示。

子程序名	返回值类型	公开	备注	
子程序1				

变量名	类型	静态	数组	备注
变量1	整数型	√		

图 3-4 定义静态变量

下面就来编写一个简单的程序, 来测试一下变量的静态属性。

第一步, 在新建的程序窗口中添加编辑框组件、按钮组件和画板组件各一个, 并将按钮组件的标题改为“连加”。

第二步, 双击按钮组件, 在“\_按钮 1\_被单击”子程序中, 新建 2 个局部变量。

第三步, 一个变量名定义为“静态变量”; 另一个变量名定义为“非静态变量”。都为整数型变量。将“静态变量”设置为静态属性。

第四步, 输入代码:





子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
静态变量	整数型	✓		
非静态变量	整数型			

静态变量 = 静态变量 + 到数值 (编辑框1. 内容)

非静态变量 = 非静态变量 + 到数值 (编辑框1. 内容)

画板1. 滚动写行 (静态变量, 非静态变量)

第五步，试运行程序，在编辑框中输入一个数，然后点击按钮。如图 3-5 所示。

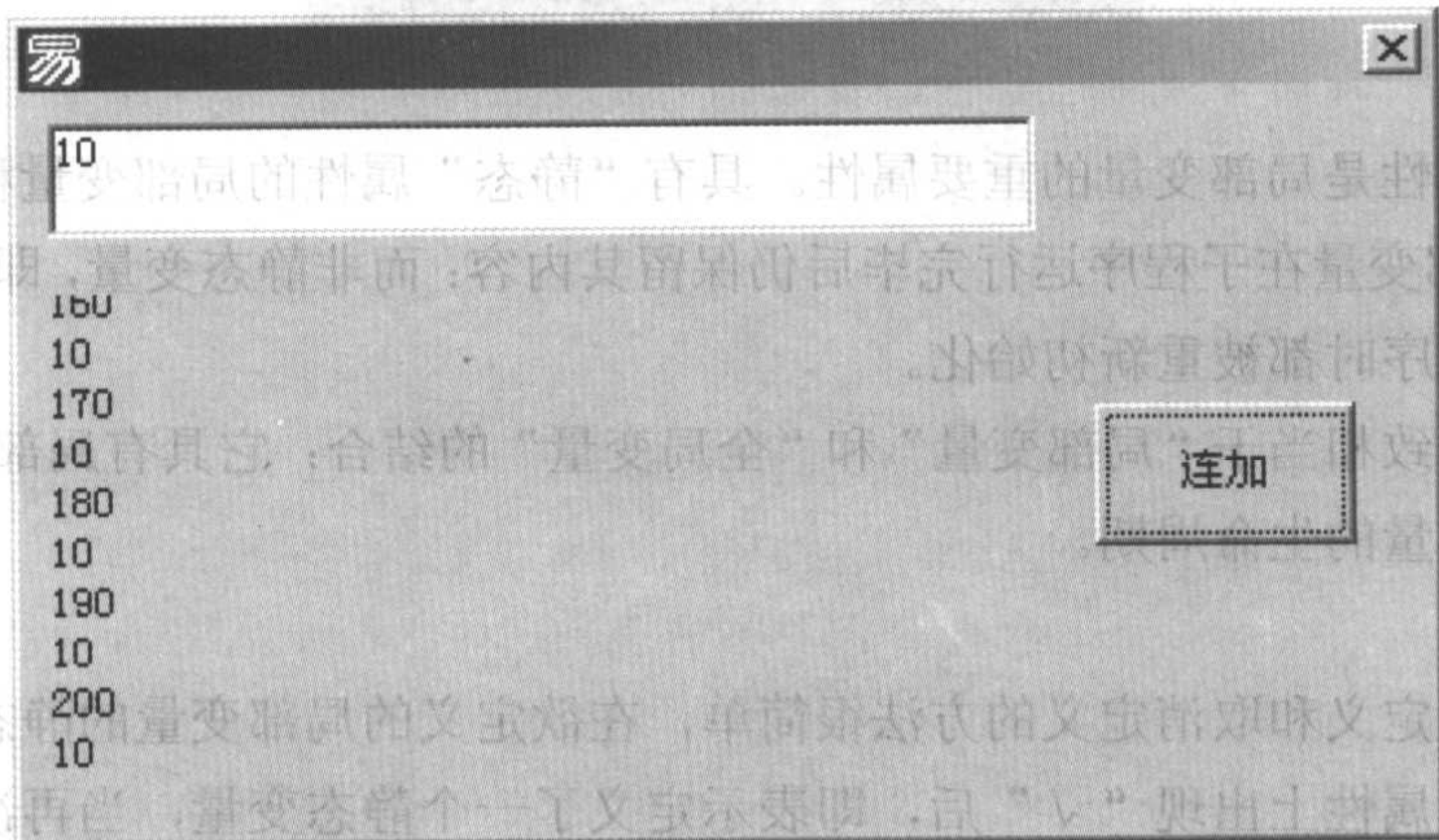


图 3-5 例程运行效果

可以看到，每次点击按钮，画板就会显示 2 行内容，其中第二行的数字是不变的，显示的是非静态变量中的数值；而第一行的数字每次都递增编辑框中的数字，显示的是静态变量中的数字。

分析代码可以发现：每次点击按钮，静态变量和非静态变量都会加上编辑框中的数。而非静态变量的值每次点击按钮后，都会恢复它的初始值，所以代码：

非静态变量+到数值 (编辑框 1. 内容)

其实等于代码：

0 + 到数值 (编辑框 1. 内容)

静态变量会保存上次的值，所以总是递增的。

### 3.3 数组变量

数组变量可以存放一组数据，数组的成员数可以自行定义，并且在程序运行后还可以



单维数组变量，成员的表示形式为：

数组变量名[数组成员下标] 变量名用大写字母，量变四字母用大写字母，成员下标用数字

音更慘狀，新資，是當，榮耀

变量[3][2]、变量[3][3]

变量[1]、变量[2]、变量[3]、变量[4]、变量[5]、变量[6]、变量[7]、变量[8]、变量[9]

数组变量也可以直接称之为数组，本书中以后如没有特别声明，所提到的数组都是指





的数组变量。

### 3. 数组变量的赋值

#### (1) 直接赋值

数组变量的赋值，就是给数组中的成员赋值，每个成员都有独立的存储空间。数组中的每个成员都可以看做是单独的变量，可以使用给变量赋值的方法来给数组的成员赋值，例如，给一个有 2 个成员的整数型数组赋值，让每个成员都为 100，程序代码为：

```
变量[1] = 100
```

```
变量[2] = 100
```

例如，给一个 2 维的整数数组赋值，每个维有 2 个成员，每个成员都赋值 100：

```
变量[1][1] = 100
```

```
变量[1][2] = 100
```

```
变量[2][1] = 100
```

```
变量[2][2] = 100
```

#### (2) 连续赋值

给数组变量赋值还有一个十分简便的方法，就是使用一对大括号将要赋予的值括起来，每个值都用“，”号隔开，被隔开的值赋予数组中的对应位置的成员，例如上面讲的给有 2 个成员的数组赋值，每个成员都赋值 100，就可以使用下面的方法：

```
变量 = {100, 100}
```

使用这种方法给数组成员很多的数组赋值，尤为方便，如给一个有 10 个成员的整数数组，就可以输入：

```
变量 = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}
```

#### (3) 命令赋值

可以使用“连续赋值( )”命令给数组赋值，“连续赋值( )”命令可将指定的常数、常数集、常量、资源、对象或者变量赋予到一系列变量或变量数组中去。参数<1>的名称为“用作赋予的值或资源”，参数<2>的名称为“被赋值的变量或变量数组”，命令参数表中最后一个参数可以被重复添加。具体也可以参考即时帮助，例程如：

```
局部变量：变量1 数据类型：整数型 数组：3
```

```
连续赋值 (100, 变量1 [1], 变量1 [2], 变量1 [3])
```

#### (4) 直接用命令的返回值给数组赋值

有些命令的返回值就是一个数组，所以可以直接使用该返回值给数组赋值。赋值的时候首先要注意，根据命令返回数组的数据类型来给数组定义数据类型；还要注意数组的成员数是可变的，并且命令返回的数组成员数也不固定，所以可以定义接收返回值的数组成员数为 0，当该数组接收了命令的返回值后，会自动定义成员数。例如：“分割文本( )”命令的返回值，就是一个文本型的数组，在程序中可以表示为：



变量名	类型	静态	数组	备注
文本数组	文本型		0	

文本数组 = 分割文本 (取运行目录 0, “\”, )

代码运行后，如果运行目录是“d:\Program Files\”，那分割后返回后子文本就是“d:”、“Program Files”、“e”，即文本数组就有了 3 个成员，每个成员的值就是返回来的子文本。

### (5) 命令参数是数组

有些命令或组件方法的参数是数组，此类命令或方法就需要根据帮助信息，来了解参数中的数组所代表的意义。例如画板的画多边形方法，第一个参数中的添入的数组顺序记录多边形各顶点的横向及纵向坐标值，坐标是成对出现，所以，数组中每 2 个成员就代表了要画出多边形的 1 个点，要让画板画出一个三角形就可以用以下代码：

变量名	类型	静态	数组	备注
三角形坐标数组	整数型		6	

三角形坐标数组 [1] = 50

三角形坐标数组 [2] = 10

三角形坐标数组 [3] = 0

三角形坐标数组 [4] = 100

三角形坐标数组 [5] = 100

三角形坐标数组 [6] = 100

画板1. 画多边形 (三角形坐标数组, )

## 3.3.2 动态管理数组变量

数组变量的最大特点就是它的成员数也是可变的，这样就有非常大的灵活性，在程序运行中，可以动态的为数组添加成员、删除成员等等，这样就达到了对数组的动态管理。

动态管理数组主要通过命令来实现，易语言提供了 10 个专门针对数组操作的命令，如图 3-8 所示。

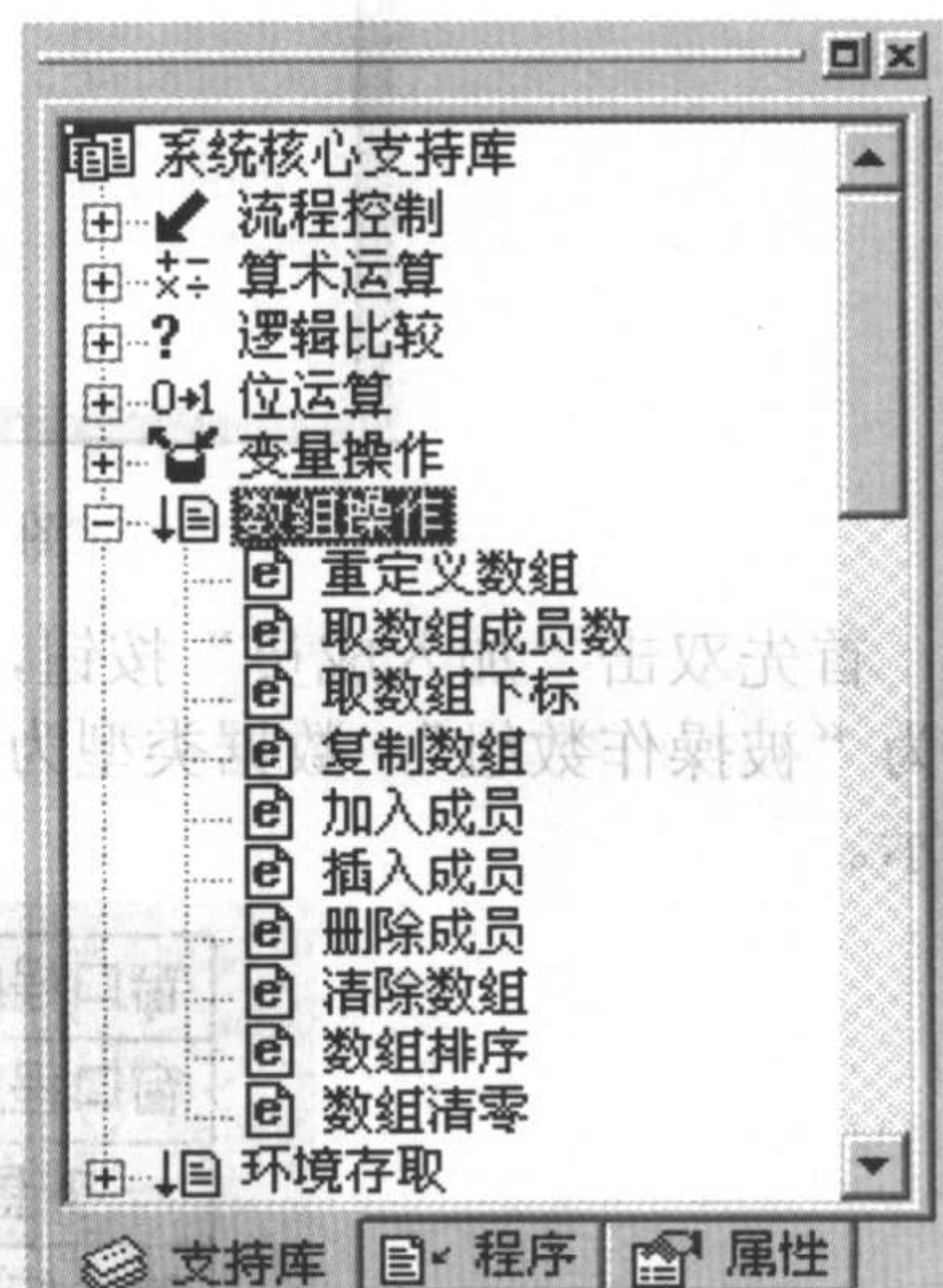


图 3-8 数组操作命令

各命令解释参见表 3-2 所示。





表 3-2 数组操作命令

命令名	命令解释
重定义数组	可以重新定义指定数组的维数及各维的上限值
取数组成员数	取指定数组变量的全部成员数目，如果该变量不为数组，返回-1
取数组下标	返回指定数组维可用的最大下标（最小下标固定为 1）
复制数组	将数组数据复制到指定的数组变量
加入成员	将数据加入到指定数组变量的尾部，自动增加其成员数目
插入成员	将数据插入到指定数组变量的指定位置，自动增加其成员数目
删除成员	删除指定数组变量中的成员，自动减少其成员数目
清除数组	删除指定数组变量中的所有成员，释放这些成员所占用的存储空间
数组排序	对指定数值数组变量内的所有数组成员进行快速排序
数组清零	将指定数值数组变量内的所有成员值全部设置为零

下面就结合实例来了解如何动态的管理数组。首先新建一个易程序，在窗口中添加 2 个编辑框组件和 1 个列表框组件，再添加 5 个按钮组件。如图 3-9 所示。

添加列表框组件是因为对数组的操作是不可见的，是在程序内部进行操作，这里将对数组的操作过程用列表框显示出来；两个编辑框一个用来填写向数组中添加的内容，另一个用来填写“插入成员”和“删除成员”的成员位置，两个编辑框的“输入方式”属性都设置成“整数文本输入”。5 个按钮组件的标题用 5 个常用的组操作命令名来命名。

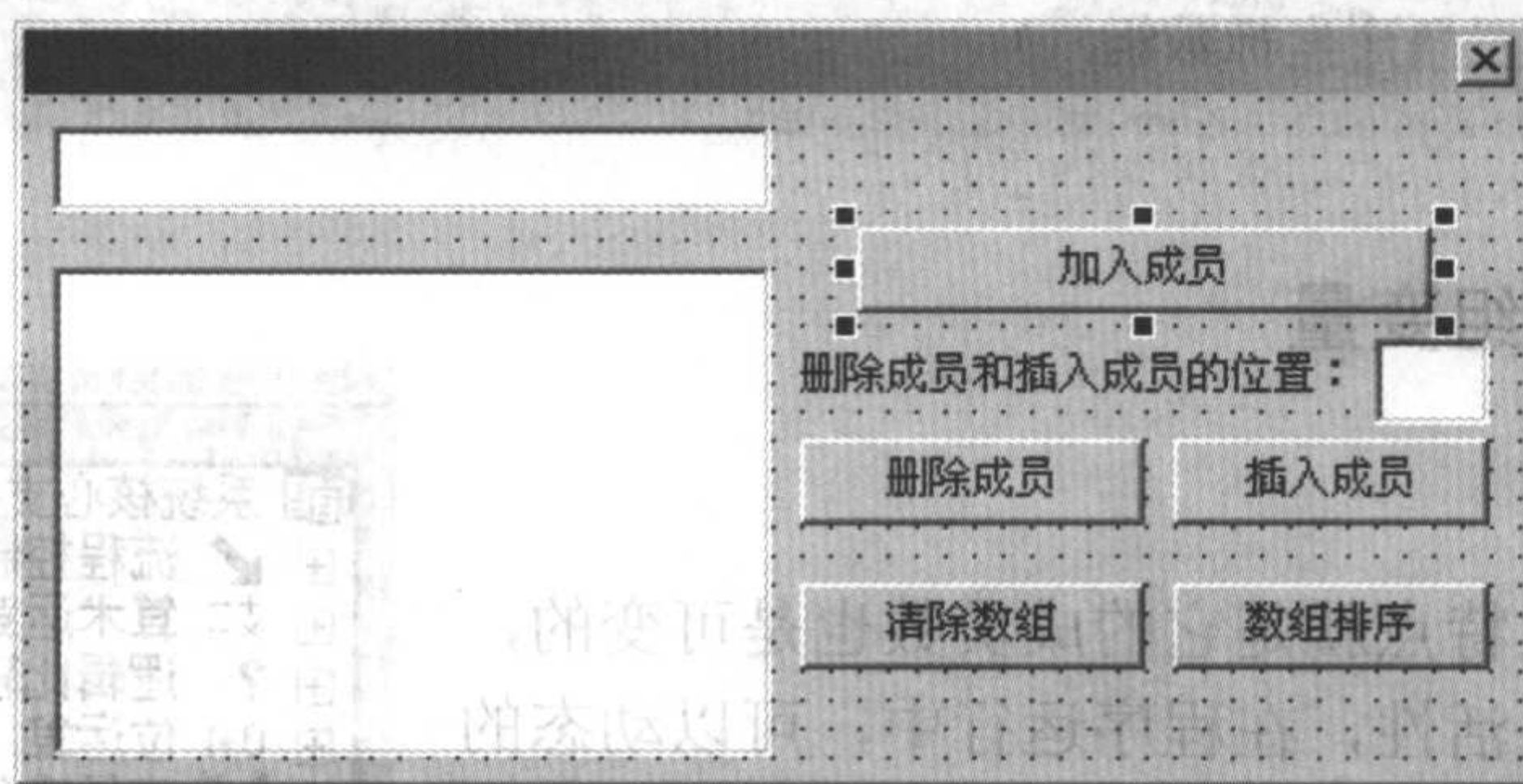


图 3-9 数组操作例程的程序界面

首先双击“加入成员”按钮，切换到程序编辑界面，然后新建一个程序集变量，变量名为“被操作数组”，数据类型为“整数型”，然后定义为 0 个成员的数组变量。如图 3-10 所示。

窗口程序集名	备 注		
窗口程序集1			
变量名	类 型	数 组	备 注
被操作数组	整数型	0	

图 3-10 例程中定义一个数组



### 1. “计次循环首 ()”和“取数组成员数 ()”命令

“计次循环首 ()”命令经常和“取数组成员数 ()”命令一起使用，用一个计次循环，就可以轻松的调用数组中的每一个成员。方法如下：首先用“取数组成员数 ()”命令将数组中的成员数取出来，然后用此成员数限定“计次循环首 ()”命令循环次数，每次循环，“循环次数变量”就会递增 1，所以就可以用：

数组名 [循环次数变量]

用这行代码就可以依次访问数组中的每个成员。在本例程中首先要考虑：要用一个列表框显示出数组中的每个成员，所以要将数组中的每个成员依次取出，并添加到列表框中，并且每次对数组操作后都要重新显示数组中的成员。新建一个子程序，这个子程序专门用来让列表框刷新显示数组内容的，这样可以节省很多代码。

在程序设计界面用 Ctrl+N 来新建一个子程序，将子程序名改为“刷新列表框”，然后在子程序中输入代码：

子程序名	返回值类型	公开	备注
刷新列表框			

变量名	类型	静态	数组	备注
循环次数变量	整数型			

列表框1.清空 ()

计次循环首 (取数组成员数 (被操作数组), 循环次数变量)

列表框1.加入项目 (到文本 (被操作数组 [循环次数变量]),)

计次循环尾 ()

程序中的计次循环，第一次循环，“循环次数变量”是 1，就将“被操作数组[1]”加入了列表框；第二次循环，“循环次数变量”是 2，就将“被操作数组[2]”加入了列表框，依此类推，就将数组中所有的成员加入了列表框。由于被操作数组成员是整数型，所以要转换成文本型后再加入列表框。

### 2. “加入成员 ()”命令

“加入成员 ()”命令很简单，第一个参数填写欲加入成员的数组名，第二个参数填写欲加入的成员值。例程中是将编辑框 1 中的内容加入数组，所以用鼠标双击“加入成员”按钮后产生的“\_按钮 1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

加入成员 (被操作数组, 到数值 (编辑框1.内容))

刷新列表框 ()

刷新列表框 () 是用来调用刚才新建的“刷新列表框”子程序，子程序被调用后就会运行子程序中的代码。以后每次对数组操作后都将调用这个子程序，用来重新显示被操作后的数组成员情况。

### 3. “插入成员 ()”命令

“插入成员 ()”命令，其实就是可以在数组中的指定位置加入成员。命令的第二个参





数填写欲插入成员的位置。例程中编辑框 2 用来填写插入成员的位置，双击“插入成员”按钮，在产生的子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮2_被单击			

插入成员 (被操作数组, 到数值 (编辑框2.内容), 到数值 (编辑框1.内容))

刷新列表框 ()

## 4. “删除成员 ()” 命令

“删除成员 ()” 命令可以将数组中指定位置的成员删除，命令的第二个参数，用来填写欲删除的成员位置。例程中用编辑框 2 来填写欲删除的成员位置，双击“删除成员”按钮，在产生的子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮3_被单击			

删除成员 (被操作数组, 到数值 (编辑框2.内容), )

刷新列表框 ()

## 5. “清除数组 ()” 和 “数组清零 ()” 命令

“清除数组 ()” 命令用来删除指定数组中的所有成员；而“数组清零 ()” 命令是将数值型数组的所有成员都设置为 0。这里要注意，“数组清零 ()” 命令只能对数值型数组进行操作。例程中只使用到了“清除数组 ()” 命令，双击“清除数组”按钮，在产生的子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮4_被单击			

清除数组 (被操作数组)

刷新列表框 ()

## 6. “数组排序 ()” 命令

数组排序命令也只能对数值型数组进行操作，可以将一个数值型数组进行从大到小或从小到大的重新排列。双击例程中的“数组排序”按钮，在产生的子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮5_被单击			

数组排序 (被操作数组, )

刷新列表框 ()

还有“重定义数组 ()”、“复制数组 ()”、“取数组下标 ()” 命令，都是对数组操作非常方便的命令，以后编程会陆续使用到，这里就不详细介绍了。参见随书光盘中的例程“数组操作.e”



## 3.4 易语言常量

### 3.4.1 了解常量

常量是一个固定的量，其值不可以被改变。易语言中规定了一些常量，这些常量都有固定的值，例如易语言中的“#蓝色”代表了数值 16711680、“#F 键”代表了数值 70，所以在程序中使用“#蓝色”其实是调用了“16711680”这个颜色值。

易语言中的核心支持库定义了许多常量，这些常量可以直接用#常量名即可调用，有数值型常量，如颜色值：#蓝色、#绿色；有文本型的常量，如#引号等等。易语言的扩展支持库也有许多常量的定义，并且新增加的支持库中，有的也会增加新的常量。

可以在易语言的支持库面板对各支持库中的定义的常量进行查询。展开一个支持库，可以在被展开项目的最后一项找到“常量”选项，点击“常量”，就可以展开该支持库中的常量列表，点击一个常量后，按下“F1”键，可以在最下面的提示框中看到有关该常量的帮助信息，并可以查到该常量的值。如图 3-11 所示。每次安装了新的支持库，也可以使用该方法来查看新支持库中引用了哪些常量，可以方便程序的编写。

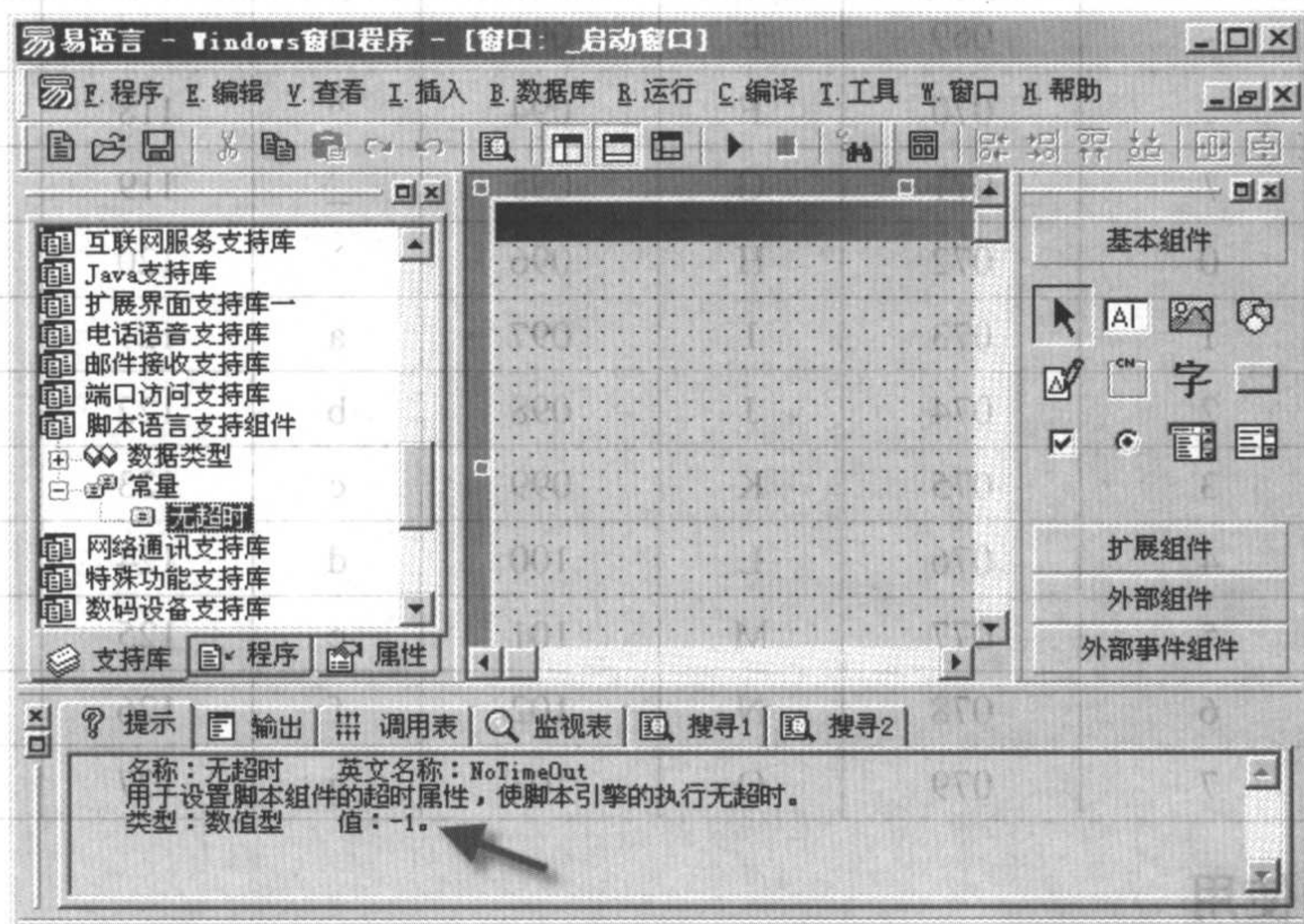


图 3-11 易语言各支持库中的常量查询

### 3.4.2 ASCII 码

ASCII 码是电脑与因特网中最普遍的文字档案格式，是各种计算机通用的一种常量。例如字符 a 的 ASCII 码是 97、字符 b 的 ASCII 码是 98 等等，参见表 3-3 所示。可以使用易语言中的“字符 ()”和“取代码 ()”命令，在 ASCII 码和字符之间进行转换。例如：

信息框 (取代码 (“a”, 1), 0, )





信息框会显示“a”的 ASCII 码。

表 3-3 常用 ASCII 码表

ASC 值	对应字符	ASC 值	对应字符	ASC 值	对应字符	ASC 值	对应字符
032	(space)	056	8	080	P	104	h
033	!	057	9	081	Q	105	i
034	"	058	:	082	R	106	j
035	#	059	;	083	S	107	k
036	\$	060	<	084	T	108	l
037	%	061	=	085	U	109	m
038	&	062	>	086	V	110	n
039	'	063	?	087	W	111	o
040	(	064	@	088	X	112	p
041	)	065	A	089	Y	113	q
042	*	066	B	090	Z	114	r
043	+	067	C	091	[	115	s
044	,	068	D	092	\	116	t
045	-	069	E	093	]	117	u
046	.	070	F	094	^	118	v
047	/	071	G	095	_	119	w
048	0	072	H	096	`	120	x
049	1	073	I	097	a	121	y
050	2	074	J	098	b	122	z
051	3	075	K	099	c	123	{
052	4	076	L	100	d	124	
053	5	077	M	101	e	125	}
054	6	078	N	102	f	126	~
055	7	079	O	103	g	127	□

### 3.4.3 常量的使用

下面就介绍一些常用常量的使用方法：

#### (1) 颜色值常量的使用

有颜色属性的组件，在颜色属性上都有一个颜色选择器，用来直接改变颜色，颜色选择器上可直接选择颜色的颜色值都作为常量提供，在调用的时候直接输入“#颜色名”即可，如：

标签 1. 背景颜色 = #天蓝

#### (2) “#换行符”的使用



一段文本尾部加入了一个“#换行符”，接在换行符后面的文本将另起一行，相当于在记事本中输入的回车键。如果能让编辑框显示一段文本并自动换行，就需要使用换行符，将“#换行符”加到欲换行文本的前面即可，如：

编辑框 1. 是否允许多行 = 真

编辑框 1. 内容 = “易语言” + #换行符 + “编程可视化”

### (3) “#引号”、“#左引号”、“#右引号”

为了不和代码中表示文本数据的引号相冲突，程序中将文本的引号作为了一个文本常量，如果能让编辑框显示出一个引号，就要使用“#引号”常量，要显示中国标点中的引号，就要使用常量“#左引号”、“#右引号”。例如：让编辑框显示出：“我爱编程！”，需要输入以下代码：

编辑框 1. 内容 = #左引号 + “我爱编程！” + #右引号

### (4) 键代码的使用

易语言中，将标准的 101 键盘上所有键的键代码都作为了核心支持库定义的常量，在程序中使用只需要输入“#”+要调用的键名，如键盘上的 F11 的键代码，在易语言中用常量表示为：#F11 键。例如，在向编辑框中输入内容的时候，想简单的屏蔽掉某个键，就可以在编辑框的“按下某键”事件子程序中输入代码：

子程序名	返回值类型	公开	备注		
_编辑框1_按下某键	逻辑型				
参数名	类型	参考	可空	数组	备注
键代码	整型				
功能键状态	整型				

--- 如果真 (键代码 = #A键)  
 ↓  
 返回 (假)

程序运行后，在编辑框中将输入不了“A”。（当然用复制粘贴的方法是可以的）

### (5) 用常量填写参数

很多命令参数填入的都是常量，如：“时间到文本”命令，此命令将指定时间转换为文本并返回。第 1 个参数为“欲转换到文本的时间”，而第 2 个参数值可以为以下常量：1、#全部转换；2、#日期部分；3、#时间部分。在填写第二个参数时，可以填写数字，也可以直接填写常量名，如：

时间到文本 ([2004 年 3 月 16 日 5 时 11 分 11 秒], #日期部分)

## 3.4.4 枚举常量及使用方法

枚举常量是一种非常方便的常量类型，它本身就是一个常量的集合，将多个常量以成员的形式，存放在一个常量中，使用的格式为

#枚举常量名. 成员名





这里要注意，枚举常量只是一种常量的表现形式，是由易语言支持库定义的常量集合，调用方法和普通常量相同，但只能由用户来调用，不能自定义。

易语言中有很多支持库中使用了枚举常量，如核心支持库中定义的“变体类型”，“变体类型”提供变体型中所能够容纳数据类型的枚举值。如表 3-4 中的某类型枚举常量可存放于变体型中，通过“变体型.取类型( )”取回当前变体型对象的数据类型。

表 3-4 变体类型常量成员及常量值

成员	描述
未知	常量值为 -1
初空	常量值为 0
数值型	常量值为 1
文本型	常量值为 2
逻辑型	常量值为 3
日期型	常量值为 4
对象型	常量值为 5
错误值型	常量值为 6
数值型数组	常量值为 7
文本型数组	常量值为 8
逻辑型数组	常量值为 9
日期型数组	常量值为 10
对象型数组	常量值为 11
错误值型数组	常量值为 12
变体型数组	常量值为 13

表 3-4 中，列出了“变体类型”常量的所有成员名及成员的常量值，在程序中，如果想调用“变体类型”常量中的“日期型”成员，该成员的常量值为 4，程序中调用该成员就等于调用了 4 这个整数，例如用信息框显示出该成员使用代码：

信息框(#变体类型.日期型, 0, )

运行后，信息框将显示 4。

这里要注意，在核心支持库中还定义了“变体型”，“变体型”和“变体类型”不同，“变体型”是一种数据类型，可以将一个变量的类型定义成“变体型”，“变体型”的变量，可以加入成员和改变成员的值；而“变体类型”是一个常量，其值只可以调用而不可以改变。

“变体型”变量的成员和值，要通过调用命令来改变。

例如程序定义一个“变体型”变量，并加入一个文本型成员，然后给该成员赋值“我爱编程”，然后用信息框显示该成员，代码如下：



变量名	类型	静态	数组	备注
变体	变体型			

变体.创建数组 (6, 1)  
 变体.赋值 (“我爱编程”, 1)  
 信息框 (变体.取文本 (1), 0, )

对“变体型”变量的操作的其他一些命令包括：清除（）、取类型（）、取数组成员数（）、取文本（）、取数值（）、取逻辑值（）、取日期（）、取对象（）、取变体型（）、赋值（）、创建数组（）。程序中使用这些命令来操作“变体型”变量。

### 3.4.5 自定义常量及使用方法

除了各支持库定义的常量以外，易语言中还可以自定义常量，自己来规定一个新的常量及其代表的值。自定义常量，可以定义一些固定值，例如后面章节会将到的 API 函数的调用，就会涉及定义一些固定的常量值的定义；编程中使用一些自定义常量还可以增加编程的灵活性，当程序中多个地方调用了某个自定义常量时，如果改变这个自定义常量的值，那这些调用该常量的地方将会自动调用改变后的新值，这样可以节省改写程序的时间。

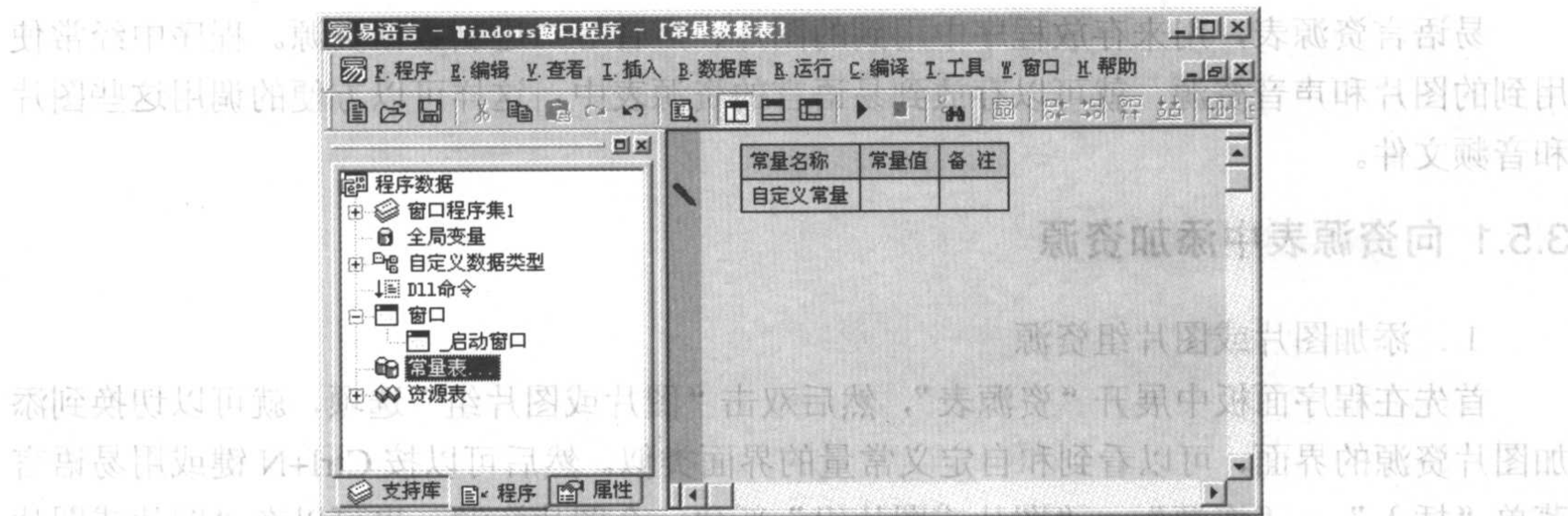


图 3-12 自定义常量

定义了一个新的常量后，可以任意定义常量的名称，然后在“常量值”上输入该常量的值。自定义常量的使用方法和非自定义的常量的使用方法相同，用“#”+自定义常量的名称。

下面就用一个简单的例程来学习使用自定义常量。新建一个易程序，在窗口中添加 1 个编辑框组件、1 个标签组件和 1 个按钮组件。然后按照上面介绍的方法自定义一个常量，常量名叫“显示内容”，然后将常量值定义为“我爱编程”。这里要说明的是，如果定义数值型的常量，直接在“常量值”上输入数值即可；如果定义文本型常量要在欲定义的文本两端加双引号。如图 3-13 所示。

常量名称	常量值	备注
显示内容	“我爱编程”	

图 3-13 定义了一个文本常量





双击窗口中的按钮，在“\_按钮1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

标签1.标题 = #显示内容

编辑框1.内容 = #显示内容

最后运行程序，按下按钮后，可以看到标签和编辑框同时显示出“我爱编程序”。可以试着在不改变代码的情况下，直接改变自定义常量的值，再次运行程序，可以看到改变常量值后，标签和编辑框显示的内容也跟着改变。

## 3.5 易语言资源表

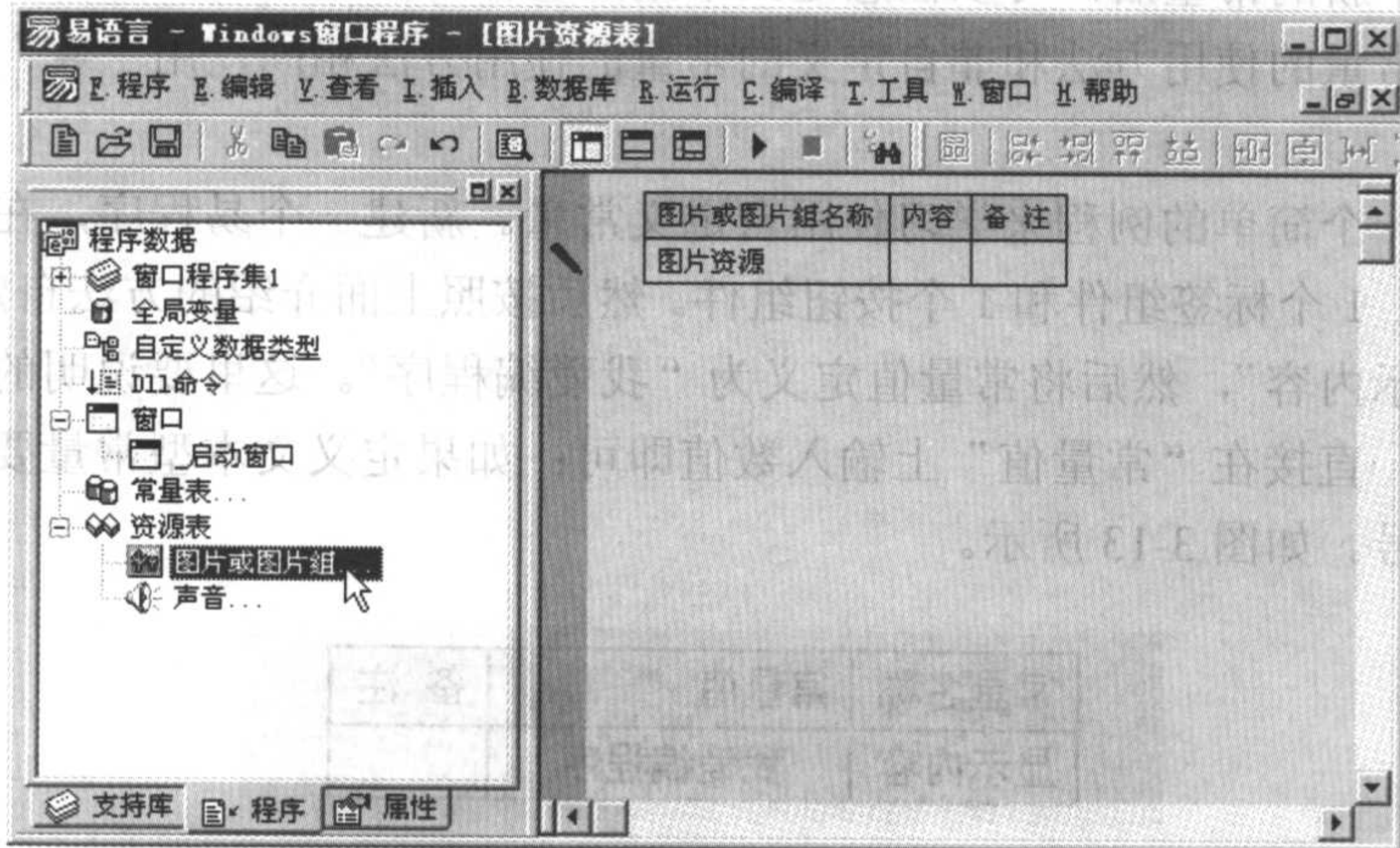
易语言资源表，用来存放程序中用到的图片、声音等二进制数据资源。程序中经常使用到的图片和声音资源，就可以存放到易语言的资源表中，这样可以方便的调用这些图片和音频文件。

### 3.5.1 向资源表中添加资源

#### 1. 添加图片或图片组资源

首先在程序面板中展开“资源表”，然后双击“图片或图片组”选项，就可以切换到添加图片资源的界面，可以看到和自定义常量的界面类似。然后可以按 **Ctrl+N** 键或用易语言菜单“插入”→“资源”→“图片或图片组”新建一个图片资源，也可以在“图片或图片组名称”上直接按回车键。

增加了一个图片资源后，可以随意定义一个名称。如图 3-14 所示。



如图 3-14 添加图片资源界面



然后在该资源的内容上双击鼠标左键或按下键盘中的任意字符键，会弹出一个“图片或图片组资源属性”对话框。可以点击对话框中的“导入新图片”按钮或“图片组...”按钮来给新的资源加入图片或图片组，加入了图片后最后要点击“加入 / 修改到程序并关闭对话框”按钮，来保存修改的内容。如图 3-15 所示。



图 3-15 给图片资源加入图片

## 2. 添加新的声音资源

添加声音资源首先双击程序面板“资源表”中的“声音...”选项，接下来的步骤和添加图片基本相同，只是最后弹出的对话框不同，可以点击对话框中的“导入新声音”按钮来加入声音资源，还可以点击“演播现行声音”按钮直接试听播放效果，声音资源中只支持“\*.MID”和“\*.WAV”格式的声音文件，最后也要点击“加入/修改到...”按钮来保存该声音文件。如图 3-16 所示。

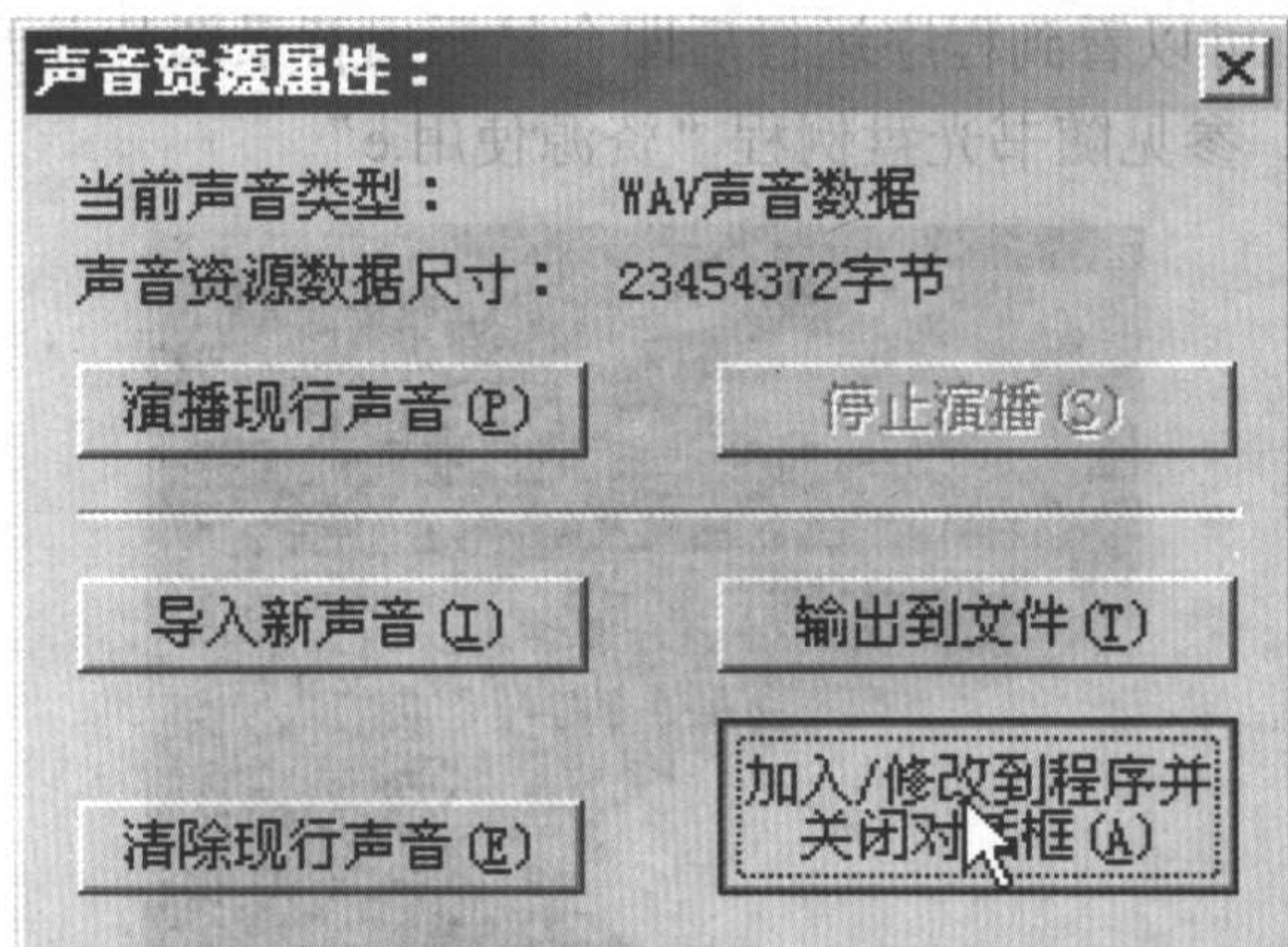


图 3-16 添加声音文件对话框





资源内容中的数值，表示了资源所占用的字节数。

### 3.5.2 使用资源表中的资源

添加后的资源都是字节集型的数据，程序中使用这些资源的方法和使用常量的方法完全相同，用“#”+“资源名”，如：

图片框 1. 图片 = # 图片资源 1

下面我们就来用资源表中的资源给窗口加上背景图片和背景音乐，首先新建一个易程序，然后按上面介绍的方法加入一个图片资源和一个声音资源，将资源名分别定为“背景图片”和“背景音乐”，并向资源中加入图片和音乐。如图 3-17 所示。

图片或图片组名称	内容	备注
背景图片	2359350	

声音名称	内容	备注
背景音乐	18052	

图 3-17 加入图片和声音资源

接下来双击启动窗口，会产生“\_启动窗口\_创建完毕”事件子程序，在该子程序中输入代码：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

```
_启动窗口.底图 = #背景图片  
_启动窗口.底图方式 = 2  
_启动窗口.背景音乐 = #背景音乐
```

最后试运行程序，可以看到程序运行后即会显示出背景图片并开始播放资源表中的音乐了。如图 3-18 所示。参见随书光盘例程“资源使用.e”

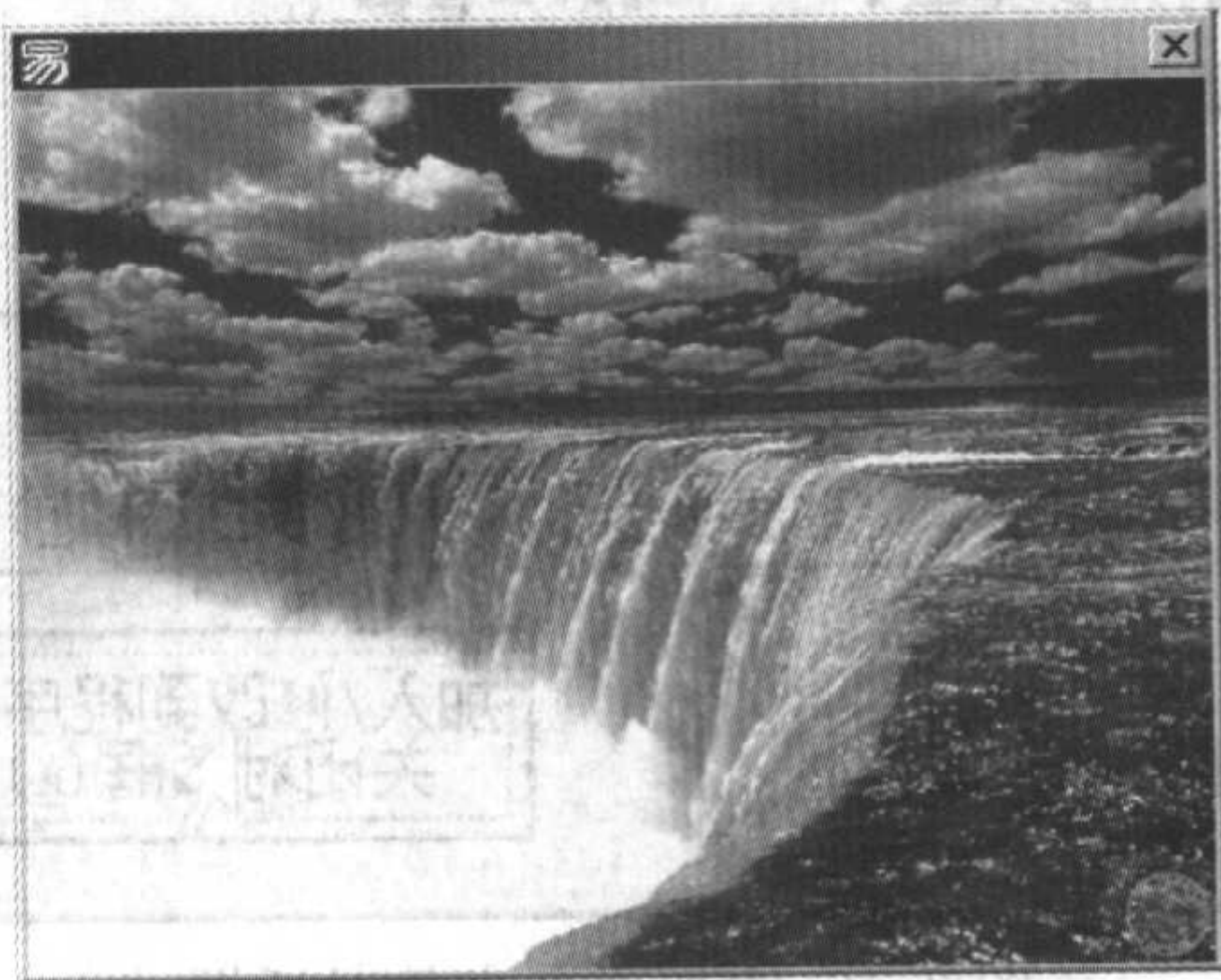


图 3-18 有背景图片和背景音乐的窗口



### 3.5.3 将资源表中的资源导出

“图片或图片组资源属性”对话框和“声音资源属性”对话框中都有“导出到文件”按钮，点击该按钮会弹出一个文件对话框，在文件名上输入欲导出的文件名和扩展名，文件的扩展名可以任意写，但一定要和该资源原文件的扩展名相同，保存后的文件才能正常打开。如果是要将别人程序中的图片导出来，就要尝试导出不同的图片扩展名，例如，试着导出“.jpg”格式或“.bmp”格式的文件，只要导出的图片可以正常打开使用即可，也可以看图片大小来判断图片的格式，“.jpg”格式的图片就相对“.bmp”文件小。导出声音资源的方法和图片相同。

还可以使用命令将资源表中的资源导出，用“写到文件（）”命令。例如：

写到文件（“C:\aa.bmp”，#图片1）

代码运行后可以将资源表中的图片资源“图片1”写到C盘根目录中的“aa.bmp”文件中保存。

### 3.5.4 向资源表中导入可执行文件

资源表中不但能导入图片和声音文件，还可以导入任意文件，如.exe的可执行文件。用图片资源表为例，方法如下：

首先在图片资源表中加入一个新资源，然后点击弹出的“图片或图片组资源属性”对话框中的“导入新图片”按钮，然后将弹出的通用对话框文件类型改为所有文件（\*.\*），并找到欲导入资源表的.exe可执行文件。如图3-19所示。



图 3-19 找到可执行文件

然后点击“打开”按钮，将该可执行文件导入到资源表。

导入到资源表的文件可以用“写到文件（）”命令写出，并用“运行（）”命令运行。例如刚导入的资源名为“执行文件资源”，可以用下面代码导出并运行：

写到文件（“c:\小计算器.exe”，#可执行文件资源）

运行（“c:\小计算器.exe”，假，）

参见随书光盘中例程“资源表中.exe资源.e”。





## 3.6 本章小结

试做以下练习：

1. 变量有自己的作用域范围。从变量的使用范围来区分，可以将变量分为\_\_\_\_\_和\_\_\_\_\_。从变量的属性来区分，还可以分为\_\_\_\_\_和\_\_\_\_\_。
2. 变量 [4] [3]，意思是一个二维变量中的第\_\_\_\_\_个成员。
3. 熟练掌握定义各种类型变量的方法。
4. 掌握查询易语言中常量值的方法，自定义几个常量，并在程序中调用。
5. 编写一个只接受键盘上 26 个英文字母输入的编辑框输入程序。
6. 用资源表中的资源给其他组件的属性赋值，如：“图片框. 图片=#图片”。

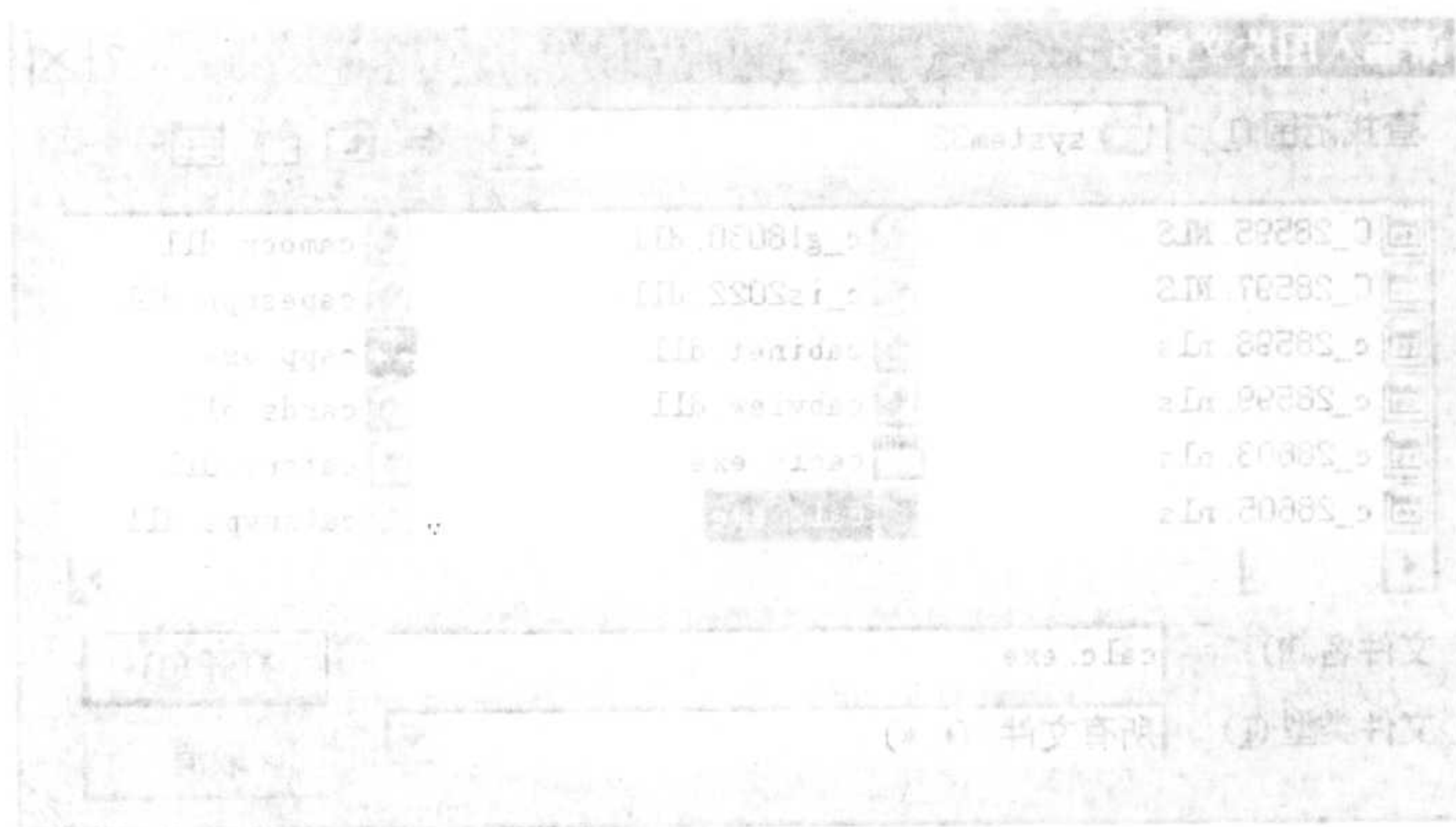


图 3-2-4 资源表

资源表是易语言中非常重要的一个概念，它用于管理程序中的各种资源。

资源表中的资源可以分为以下几类：

1. 图标资源：用于显示程序的图标。

2. 光标资源：用于显示程序的光标。

3. 声音资源：用于播放程序的声音。

4. 字符串资源：用于存储程序中的字符串。



## 第四章 常用命令

程序是由各种命令组合而成的，不同的命令完成不同的工作。易语言中提供了大量的命令，用户可以使用这些命令来实现预想的运行效果。一个程序可以实现了一种或多种功能，而这些功能的实现都离不开程序内部调用的大量命令。命令是程序的基本组成部分，要学习易程序的编写，首先就要了解易语言所提供的命令。

### 4.1 了解易语言命令

#### 4.1.1 命令的格式

易语言中的命令格式如下：

**命令名称 (参数, ...)**

大部分命令都需要填写参数，参数用括号括起来的，并用逗号分隔。部分命令无需参数，但括号不能省略，如“结束( )”命令。各种命令所要求参数的个数以及数据类型各有不同，由其语法所规定。例如“到文本( )”命令，该命令只需一个参数，参数内容为欲转换成文本的数据。有些命令的参数很多，如“子文本替换( )”命令的格式如下：

**子文本替换 (欲被替换的文本, 欲被替换的子文本, [用作替换的子文本], [进行替换的起始位置], [替换进行的次数], 是否区分大小写)**

#### 4.1.2 即时帮助和帮助文档

命令有这么多参数，强行记忆会比较困难。

当编写命令时，按 ALT+右键（方向键）可自动展开该命令的参数，以便于用户书写程序代码。如：

```

» 画板1.定位写出 (画板1.宽度 ÷ 2, 画板1.高度 ÷ 2, “我爱学习易语言!”)
»     ↳ ※横向写出位置: 画板1.宽度 ÷ 2
»     ↳ ※纵向写出位置: 画板1.高度 ÷ 2
»     ↳ ※欲写出数据: “我爱学习易语言!”

```

易语言提供了即时帮助功能和内容丰富的帮助文档。用户根本不需要将全部的命令语法和参数含义都背下来。在实际开发工作中，可以使用命令分步输入方法（在当前代码行上按“右光标键”展开），根据参数提示输入，或按 F1 键查看易语言的即时帮助，或查阅帮助文档。





在易语言的帮助系统中，所有的命令都有明确的分类。例如，想对一段文本进行操作，就要查找“文本操作”分类中的相关命令；如果想对一个文件进行操作，就要查找“文件读写”分类中的相关命令。

在易语言的支持库面板中，双击展开任意一个支持库名称，可以查到该支持库的所有命令分类；双击展开其中任意一个分类名称，可以看到属于该分类的所有命令；点击任意一个命令名称，就可以在状态夹的提示面板中看到该命令的即时帮助信息。如图 4-1 所示。

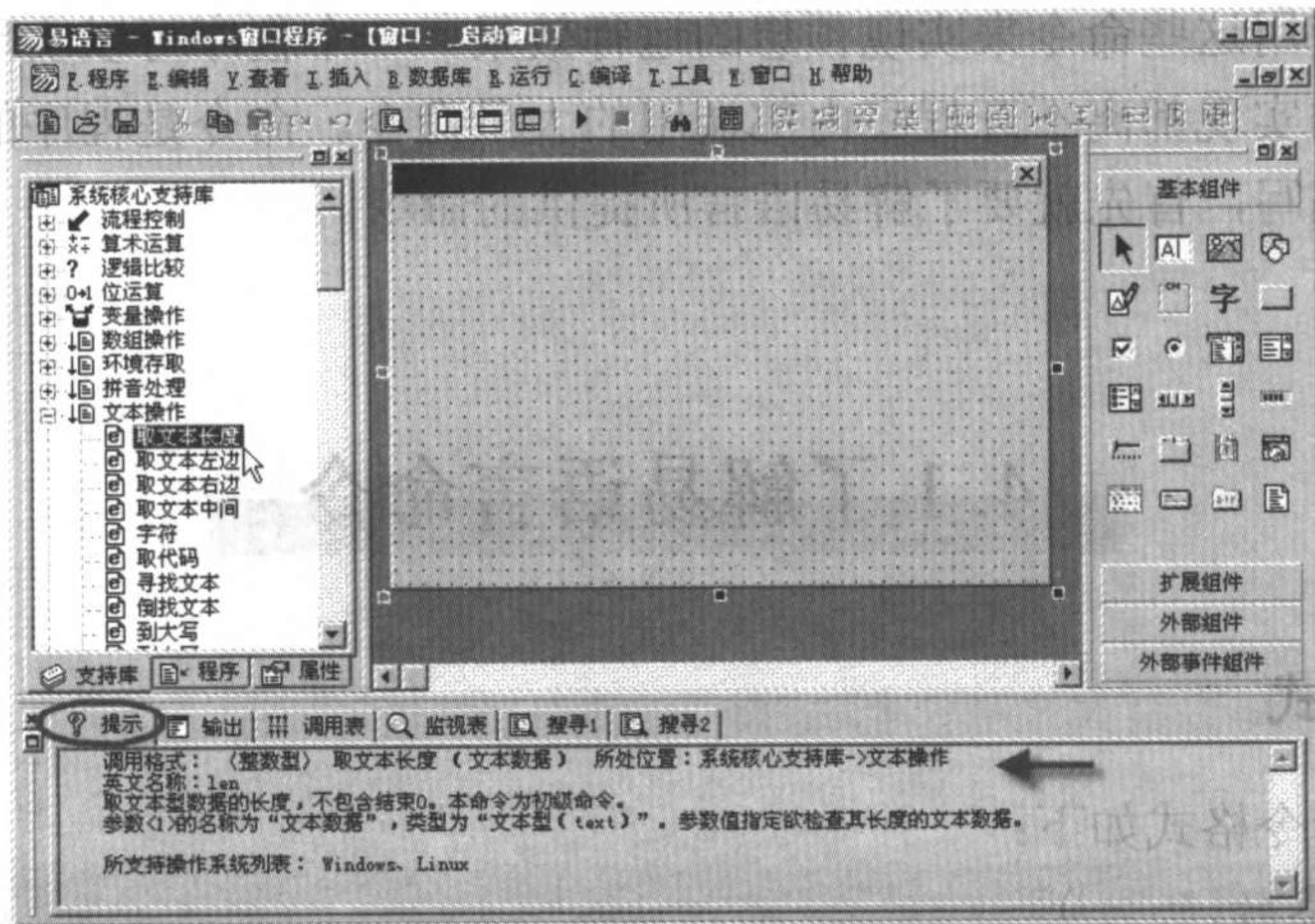


图 4-1 查看即时帮助信息

在编写程序时，如果忘记了某个命令参数的含义，可以将光标停在该命令上，然后按 F1 键，也可以在提示面板中看到该命令的即时帮助信息。

点击菜单“帮助”→“易语言知识库”，可以查看完整的帮助文档。其内容相对即时帮助更加丰富，并且每个命令都有简单的例程，查找命令也非常方便。如图 4-2 所示。

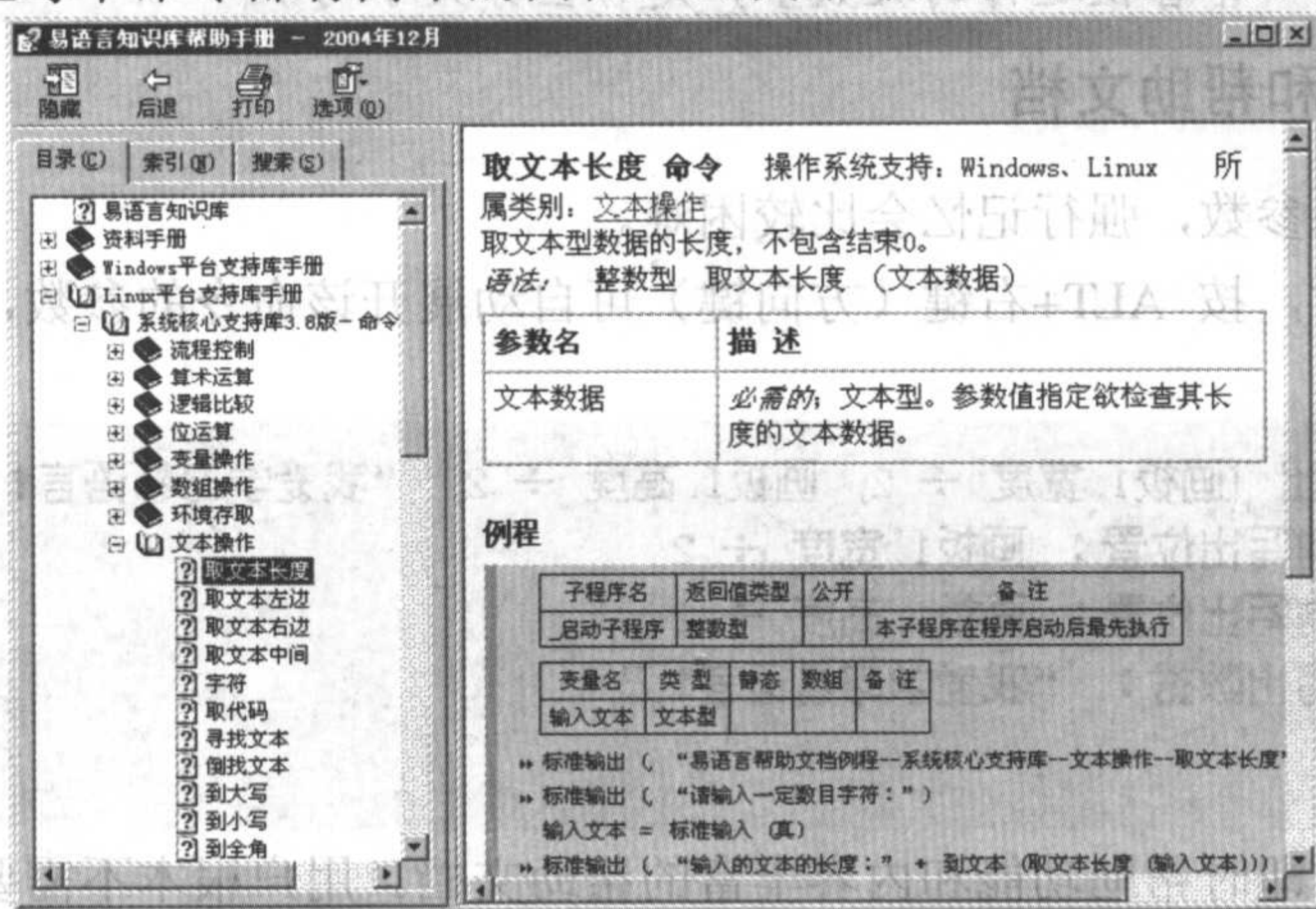


图 4-2 易语言帮助文档

易语言核心支持库中已经提供了六百多个基本命令，各扩展支持库中也包含了很多命



令。在平时的学习和开发中要注意活学活用，不要死记硬背，多使用分步输入法和即时帮助这两个功能来帮助记忆，时间长了自然就记熟了。

### 4.1.3 命令的返回值

大多数命令执行完毕后都有返回值。有的命令返回运算结果，如“求正弦（）”命令，返回求得的正弦值；有的命令返回的执行的結果，如“取文本左边（）”命令，返回取出来的文本内容；有的命令返回运行是否成功的状态，如“创建目录（）”命令，创建成功则返回“真”，创建失败则返回“假”等等。大部分时候，当前命令的返回值对后续命令来说非常重要。例如一个命令如果运行成功了，就弹出信息框提示成功，否则提示失败，如下面的代码：

```

--- 如果 (创建目录 ("c:\下载中心") = 真)
--- 信息框 ("创建成功", 0, )
--- 信息框 ("创建失败", 0, )

```

各命令的语法规定了其返回值的数据类型，在实际使用中，应当注意有可能需要对返回值的数据类型加以转换，例如，编辑框的内容属性只接收文本型，因此要显示一个数字就可以使用“到文本（）”命令将数字转换为文本形式显示，代码如下：

```
编辑框 1. 内容 = 到文本 (求平方根 (100))
```

“求平方根（）”命令的返回值是一个数值型的，如果要以文本方式显示在编辑框中，就要用“到文本（）”命令进行转换。

有些命令的返回值是一个通用型的数据，代表根据参数不同，其返回值数据类型也可以不同。例如“多项选择（，）”命令。该命令有 2 个参数，第一个参数是索引值，第二个参数是待选项，待选项可以重复添加。待选项数据类型是通用型（表示参数 2 可以是任意数据类型）的，返回哪个待选项取决于第一个参数的索引值。索引值是 1 则返回第一个待选项；索引值是 2 则返回第二个待选项。所以，所选项是哪种类型的数据，返回值就为哪种类型的数据。下面就编写一个小程序来了解一下多项选择命令。

第一步，新建一个易程序，在窗口中添加一个画板组件和一个按钮组件。

第二步，双击按钮组件，在“\_按钮 1\_被单击”事件子程序中输入如下代码：

子程序名	返回值类型	公开	备注
_按钮 1_被单击			

```
画板 1. 滚动写行 (多项选择 (4, "我爱易语言", 3.14159, 100000, [2004年11月3日], { 12345 })))
```

第三步，试运行程序，点击按钮，可以看到画板中显示出了第 4 个待选项日期时间型的[2004 年 11 月 3 日]。

有些命令无返回值，如“销毁（）”命令，此类无返回值的命令运行后不返回任何值，所以直接使用即可，例如：





子程序名	返回值类型	公开	备注
按钮2_被单击			

销毁 0

命令是否有返回值，返回值是什么数据类型，都可以通过易语言的即时帮助系统查找，在程序编辑界面，将光标停在欲查询的命令上，然后按下 F1 键，可以在提示面板中看到该命令的帮助。在提示面板中“调用格式”一行，写在命令名前面的就是该命令的返回值类型，如果无返回值则显示无返回值。如图 4-3 所示。

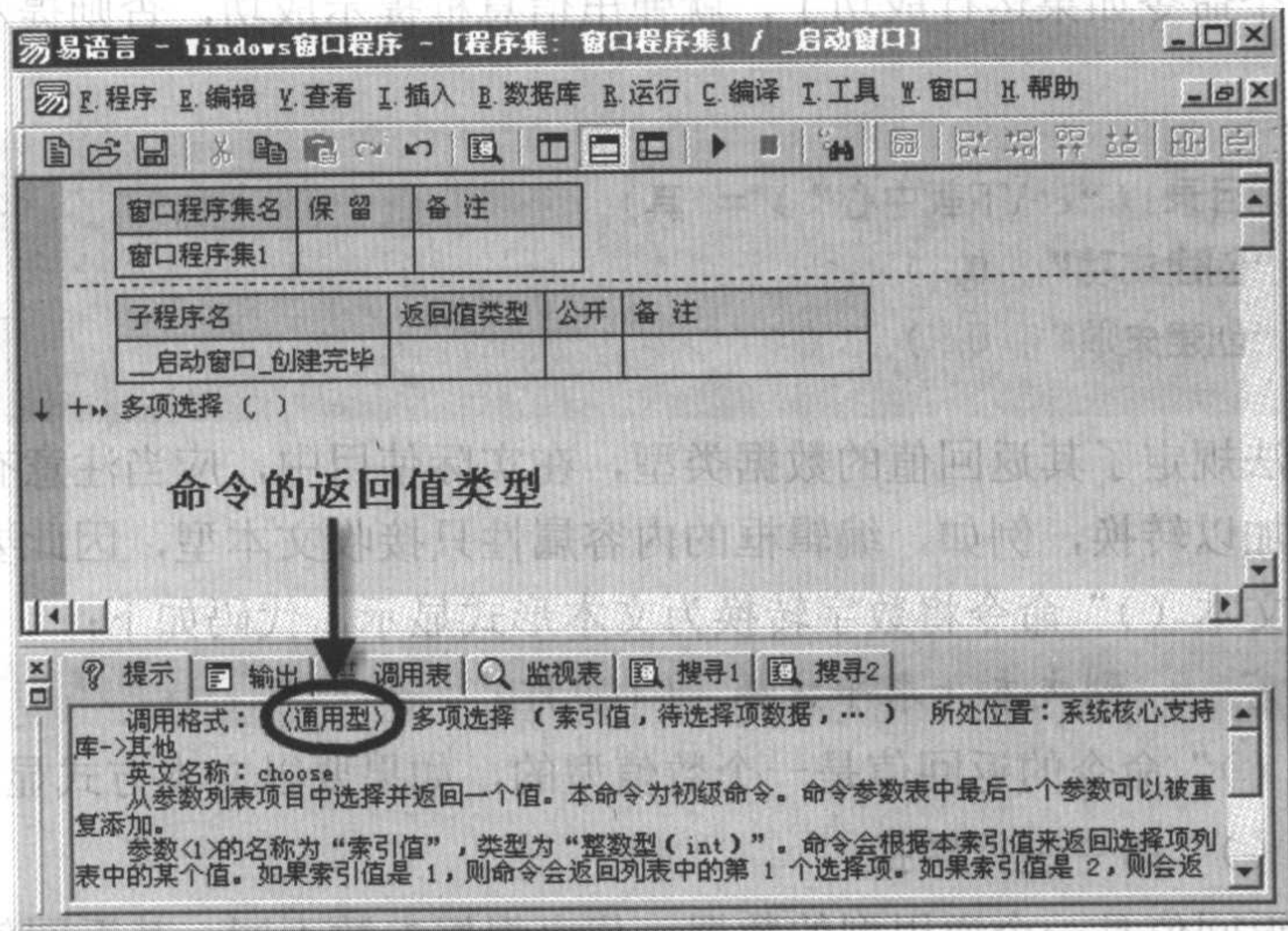


图 4-3 查看命令的返回值类型

## 4.1.4 命令的套用

易语言中的命令可以套用，即命令中包含命令。其实前面的章节已经使用过，如：

编辑框 1. 内容 = 到文本 (到数值 (编辑框 1. 内容) + 1)

此行代码在“到文本 ()”命令中套用了到数值命令，命令的嵌套调用是将一个命令的返回值作为另一个命令的参数。上面的代码首先会将编辑框中的内容转换成数值型，然后将转换后的数值加 1，将加 1 的结果转换成文本重新显示在编辑框中。

例如：“写到文件 ()”命令的参数中套用多个命令代码如下：

```
-- 如果 (写到文件 (取运行目录 () + "a.txt", 到字节集 (编辑框1.内容)))
-- 信息框 ("保存成功", 0, )
-- 信息框 ("保存失败", 0, )
```

上面的代码是将编辑框中的内容保存为程序运行目录下的一个文本文件，弹出信息框提示是否保存成功。



### 4.1.5 数组类型的参数与返回值

#### 1. 数组型参数

有些命令的参数必须是一个数组型变量，如“重定义数组（）”命令，本命令可以重新定义指定数组的维数及各维的上限值。该命令的第一个参数必须指定一个数组型变量。系统支持库中数组操作类命令的第一个参数，都要求填写欲操作的数组名，如：

变量名	类型	静态	数组	备注
数组变量	整数型		0	

» 重定义数组 (数组变量, 假, 5)

信息框 (取数组成员数 (数组变量), 0, )

加入成员 (数组变量, 100)

有些命令的参数可以填写数组或非数组的数据，如“播放 mp3（）”命令，该命令的第二个参数就可以填入一个数组或非数组的数据，并且最后一个参数可以重复添加，这样就产生了下面两种运行效果完全相同的命令。

第一种：

» 播放MP3 (1, “c:\My music\01.mp3”, “c:\My music\02.mp3”, “c:\My music\03.mp3”, “c:\My music\04.mp3”)

第二种：

变量名	类型	静态	数组	备注
播放列表变量	文本型		4	

播放列表变量 = { “c:\My music\01.mp3”, “c:\My music\02.mp3”, “c:\My music\03.mp3”, “c:\My music\04.mp3” }

» 播放MP3 (1, 播放列表变量)

这两种命令参数赋值的方式各有优点。对于数组型参数可以使用前面章节介绍的动态管理数组的方法来管理。

#### 2. 数组型返回值

有些命令的返回值是一个数组型变量，例如：“分割文本（）”命令。此类命令需要一个数组型变量存放命令的返回值。如：

变量名	类型	静态	数组	备注
分割后文本	文本型		0	

分割后文本 = 分割文本 (编辑框1. 内容, “,”, )

“分割后文本”数组变量存放分割文本命令所返回的文本型数组内容。此类命令会自动重定义数组维数(个数)，所以将变量的数组属性设置为 0 即可。分割后的文本可以用一个循环将所有子文本显示出来，代码如下：





```
---▶ 计次循环首 (取数组成员数 (分割后文本), 循环次数变量)
    编辑框1.内容 = 编辑框1.内容 + 分割后文本 [循环次数变量] + #换行符
--- 计次循环尾 ()
```

代码中用到的“计次循环首 ()”命令会在下面讲流程控制类命令时介绍。

## 4.2 流程控制命令

### 4.2.1 了解流程控制类命令

流程控制类命令在易语言中是非常重要的一类命令，可以控制程序运行的路线，例如在满足一定条件时运行哪些代码，在不满足条件时运行哪些代码等。大多数程序的编写都离不开这类命令。

流程控制命令分为 3 类：分支类流程控制命令、循环类流程控制命令和跳转类流程控制命令。

分支类流程控制命令包括：如果 ()、如果真 ()、判断 ()。

循环类流程控制命令包括：判断循环首 ()、循环判断首 ()、计次循环首 ()、变量循环首 ()。

跳转类流程控制命令包括：到循环尾 ()、跳出循环 ()、返回 ()、结束 ()。

**注意：**有时写了一段流程命令程序代码，发现可以更加优化流程命令，或用其他流程命令代替，如果这时重写，将会浪费很多时间。易语言提供“流程转换”功能，可以在各流程命令之间转换。如：写“如果 ()”命令时，如果条件只有一个，可以用“如果真 ()”代替，这时就可以转换。在流程控制类命令中，除跳转类流程控制命令，都可以在流程控制语句上使用右键弹出菜单来互相转换。在菜单中选择“转换为”→欲转换的流程控制语句。有些代码转换后运行结果相同，例如，“如果 ()”命令和单个“判断 ()”命令可互相转换。

易语言提供程序流程线，所以能很清楚的观察流程控制类命令的运行路线。例如“如果 ()”命令，当“如果 ()”命令中的条件成立，则运行“如果 ()”命令的下一行语句；当“如果 ()”命令的条件不成立，则会运行“如果 ()”命令左边箭头所指向的代码。如图 4-4 所示。

```
--- 如果 (变量1 > 变量2)
    编辑框1.内容 = 到文本 (变量1)
    编辑框1.内容 = 到文本 (变量2)
```

图 4-4 如果命令流程线



## 4.2.2 分支类流程控制命令

## 1. “如果 ( )” 命令

“如果 ( )” 命令的参数“条件”为一个逻辑型数据，非真即假。若条件为真，则程序顺序执行后续代码；若条件为假，则程序跳转到左箭头所示的代码行继续运行。

看一段例程。新建一个易程序，在“\_启动窗口”中添加 1 个编辑框组件和一个按钮组件。然后双击按钮，在“\_按钮 1\_被单击”子程序中输入代码：

```

--- 如果 (编辑框1.内容 = "")
--- 编辑框1.内容 = "你好易语言"
--- 编辑框1.内容 = "我爱易语言"

```

运行程序。此时编辑框中没有任何内容，单击按钮，可以看到编辑框显示“你好易语言”；再单击按钮，此时编辑框中已有内容，则编辑框显示“我爱易语言”。

“如果 ( )” 命令参数中还可以填写多个条件，用“或”和“且”连接，例如：

```

--- 如果 (变量1 = 100 或 变量1 = 50 或 变量1 = 25)
--- 变量1 = 0
--- 变量1 = 变量1 × 5

```

用“或”连接多个条件时，只要有一个条件成立时，整个条件参数就为真。上述代码中，当变量 1 等于 100、50 或 25 时，变量 1 的值就被改为 0，否则就等于它自身的五倍。

```

--- 如果 (变量1 = 100 且 变量2 = 50)
--- 变量1 = 0
--- 变量2 = 变量2 + 1

```

用“且”连接的多个条件，必须所有条件都成立时，整个条件参数才为真。上述代码中，只有当变量 1 等于 100，并且变量 2 等于 50 时，变量 1 的值被改为 0，否则变量 2 会自动加 1。

## 2. “如果真 ( )” 命令

“如果真 ( )” 命令从流程线上就可以看出与“如果 ( )” 命令的不同，“如果真 ( )” 命令在条件成立的时候运行“如果真 ( )” 命令下的代码，否则“如果真 ( )” 命令没有任何动作。例如：

```

--- 如果真 (变量1 > 100)
--- 变量1 = 0

```

当变量 1 大于 100，则会运行“变量 1=0”的代码，否则直接跳到判断结束后的代码继续运行。

## 3. “判断 ( )” 命令

本命令根据提供的逻辑参数的值，来决定是否改变程序的执行位置，如果提供的逻辑参数值为“真”，程序继续顺序向下执行，否则跳转到下一分支处去继续判断。





“判断 ( )”命令主要用于条件的分支选择，如下面 2 段代码运行效果相同，程序结构也相同，但使用“判断 ( )”命令，代码流程结构要清晰许多，而使用“如果 ( )”命令，不仅会使程序嵌套太多，程序代码难以看清楚，也降低了程序运行效率。

用“如果 ( )”命令编写的代码：

```
如果 (取文本右边 (取运行目录 (), 1) = "\")
    目录 = 取运行目录 ()
    如果 (目录 = "")
        目录 = 取运行目录 () + "\"
```

用“判断 ( )”命令编写的代码：

```
判断 (取文本右边 (取运行目录 (), 1) = "\")
    目录 = 取运行目录 ()
    判断 (目录 = "")
        目录 = 取运行目录 () + "\"
```

现在用“判断 ( )”命令编写一个做单选题的程序。首先，新建一个 Windows 程序，然后在“\_启动窗口”中添加 4 个单选框组件，1 个按钮组件和 1 个标签组件，标签组件的标题属性输入单选题的题目，4 个单选框组件的标题属性，分别输入单选题的备选项，按钮组件的标题输入“答题”。如图 4-5 所示。



图 4-5 判断题程序界面

其次，双击“答题”按钮，在“\_按钮 1\_被单击”子程序中输入代码，用来判断选择是否正确：



子程序名	返回值类型	公开	备注
_按钮1_被单击			

```

--- 判断 (单选框1.选中 = 真)
--- 信息框 ("回答错误,请继续努力", #错误图标, )
▶ 判断 (单选框2.选中 = 真)
--- 信息框 ("回答错误,请继续努力", #错误图标, )
▶ 判断 (单选框3.选中 = 真)
--- 信息框 ("回答错误,请继续努力", #错误图标, )
▶ 判断 (单选框4.选中 = 真)
--- 信息框 ("恭喜! 回答正确", #信息图标, )
--- 信息框 ("请选择一个答案", #询问图标, )
    
```

运行程序。当按钮被单击事件触发后，程序开始判断哪个单选框被选中。如果正确答案所在单选框被选中，则弹出“回答正确”的信息框；选择其他单选框会弹出“回答错误”信息框；如果没有单选框被选中，则弹出“请选择”信息框。如图 4-6 所示。

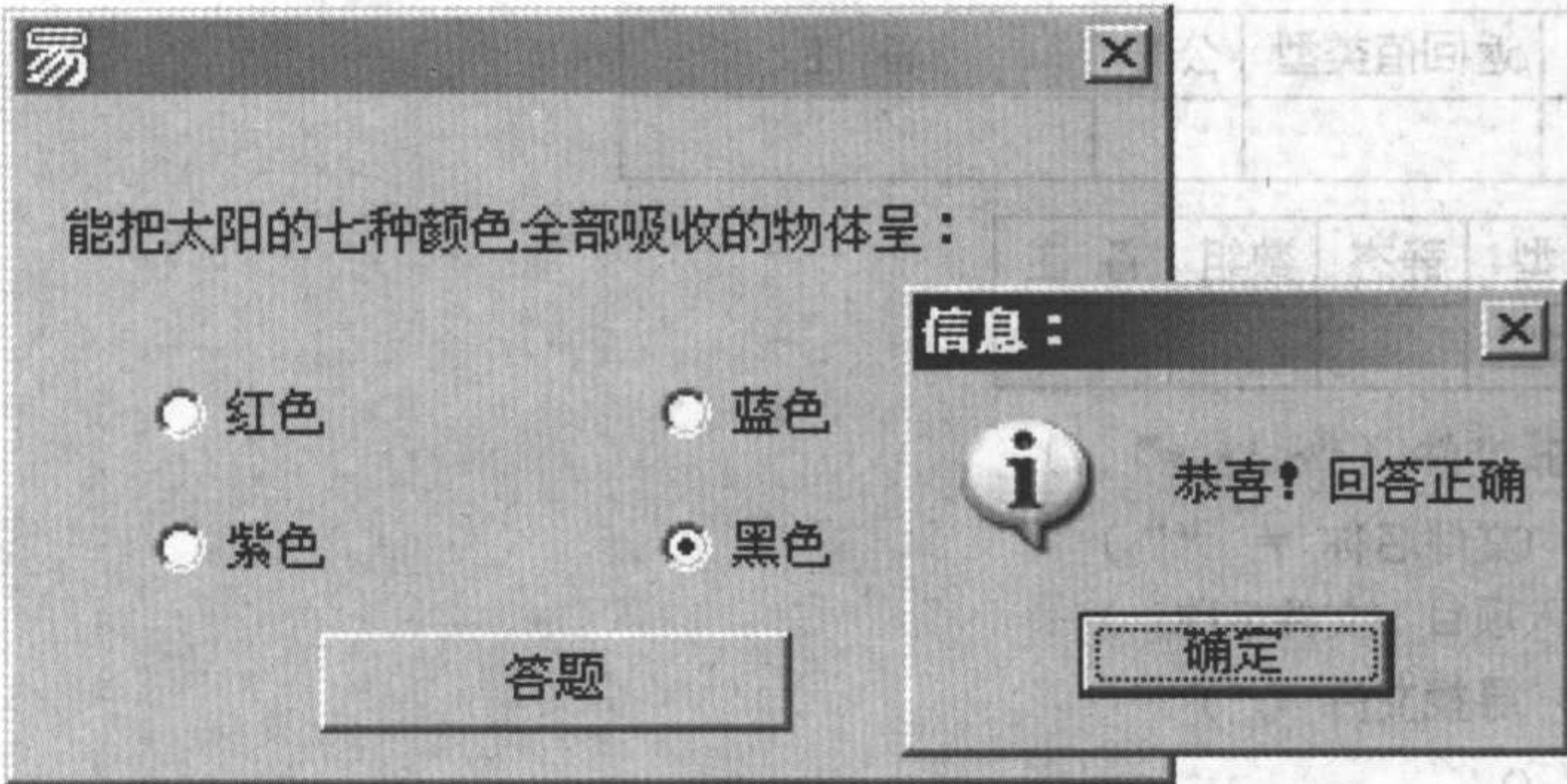


图 4-6 选择题例程运行效果

本例程源码见随书光盘中本章例程“判断命令应用.e”

### 4.2.3 循环类流程控制命令

循环类流程控制命令可以在一定条件下多次执行重复的代码。例如，将某数据库中前 100 条记录的“姓名”字段内容读出并显示在表格中，使用循环命令只需要几行代码即可实现。

循环流程类命令都由循环首和循环尾 2 部分组成，输入了循环首命令，循环尾就自动出现。循环首表示循环的开始，循环尾表示循环的结束，循环首和循环尾之间的代码，是循环类命令要重复执行的代码。

#### 1. “判断循环首 ()”和“循环判断首 ()”命令

“判断循环首 ()”命令首先检查判断条件是否成立。如果不成立，直接跳到循环尾后的代码继续执行；如果条件成立，则进入循环。每次循环结束后，会再一次检查“判断循环首”中的条件，如果条件不成立了，就退出循环，执行后续代码。例如：让画板滚动





写出 100 以内的偶数，代码如下：

变量名	类型	静态	数组	备注
变量1	整数型			

```
--> 判断循环首 (变量1 < 100)
    变量1 = 变量1 + 2
--> 画板1.滚动写行 (变量1)
--- 判断循环尾 ()
```

“循环判断首( )”命令是先循环再判断，即首先运行一次循环首和循环尾之间的代码，再判断条件是否成立。如果“循环判断尾( )”中的条件为真，就跳到循环首处继续循环，如果条件不成立，则循环终止。将上面的代码中的“判断循环首”直接转换为“循环判断首”，运行结果是相同的。可以看出，这两个命令在一定情况下是可以互换的，但由于两个命令的判断位置不同，有可能对循环体内的运行结果造成影响，在实际应用中要注意区分。

下面编写用列表框组件列出 C 盘根目录下所有文件的程序。

首先，新建一个 Windows 窗口程序，然后添加 1 个列表框组件和 1 个按钮组件。

双击按钮组件，在“\_按钮1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
文件名称	文本型			

```
文件名称 = 寻找文件 (“c:\*.*”, )
--> 判断循环首 (文件名称 ≠ “”)
    列表框1.加入项目 (文件名称)
    文件名称 = 寻找文件 ( )
--- 判断循环尾 ()
```

这里用到了“寻找文件( )”命令，当使用“寻找文件( )”命令在指定目录连续检索相同条件的文件（非子目录）时，第二次调用无需填写参数，该命令会自动继续向下寻找。

运行程序，程序会将 C 盘根目录中所有的文件，包括被隐藏文件和系统文件都显示在列表框中。如图 4-7 所示。

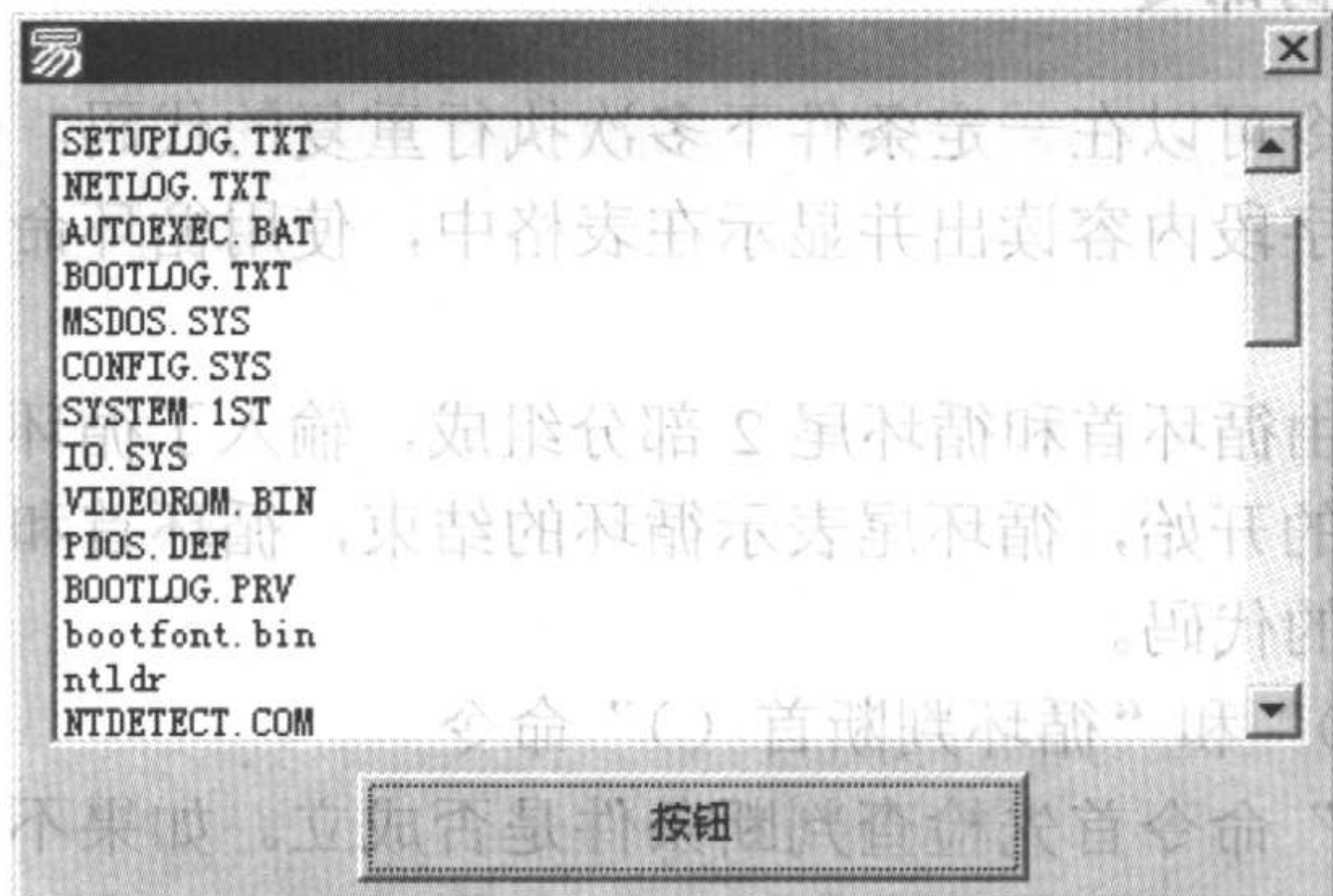


图 4-7 列表框列出 C 盘根目录的所有文件



## 2. “计次循环首( )”命令

令的第二个参数可以填入一个变量，

所有整数相加的程序。

“启动窗口”中添加一个编辑框组件和一个按钮组件。

式”属性设置为“整数文本输入”，双击按钮，在“\_按钮

变量名	类 型	静态	数组	备 注
循环次数变量	整型			
相加变量	整型			

→ 计次循环首 (到数值 (编辑框1. 内容), 循环次数变量)

$$\text{相加变量} = \text{相加变量} + \text{循环次数变量}$$

```

--- 计次循环尾 0

```

编辑框1.内容 = 到文本 (相加变量)

最后，试运行程序，在编辑框中输入一个大于 1 的整数，然后点击按钮，可以求出 1 到编辑框中的数值范围以内所有整数的和。但要注意的是，如果输入的数过大，就会导致循环的时间过长，程序暂时失去响应；如果输入的数值范围太大，可能会出现数据溢出情况。

参数, 规定了变量的起始值, 目标值和递增值,

好 友 記 實

变量名	类 型	静态	数组	备 注
变量1	整数型			
相加变量	整数型			

→ 变量循环首 (100, 200, 1, 变量1)

相加变量 = 相加变量 + 变量1

--- 变量循环尾 0 (鱼皮翻书) 本文档由 鱼皮翻书 编写, 如有不妥, 请指正。

编辑框1.内容 = 到文本 (相加变量)

循环结束后的“相加变量”就是求得的结果。最后将结果显示在编辑框中。

单的数学题:





如果设每只母鸡值 5 元，每只公鸡值 3 元，三只小鸡值 1 元。现在用 100 元钱买 100 只鸡，问能买母鸡、公鸡、小鸡各多少只？

这实际上是一个不定方程的问题：有三个未知数，却只有两个方程式。可想而知，答案应该不止有一种。

设母鸡、公鸡、小鸡数分别为 I, J, K，则应满足如下条件：

$$\begin{cases} I+J+K=100 \\ I*5+J*3+K/3=100 \end{cases}$$

这道题需要通过枚举法来求解。枚举法的基本思想是根据提出的问题，列举所有可能情况，并用问题中提出的条件检验哪些是不需要的。

枚举法的特点是算法比较简单，但当枚举的可能的情况较多时，执行枚举算法的工作量将会很大。

本题就要枚举出母鸡、公鸡和小鸡所有可能出现的数量，然后套用上面的两个方程式，如果符合 2 个方程式，就是一种答案。

可以使用“计次循环首 ( )”命令，列举出所有可能出现的组合情况。

第一步，新建一个易程序，在“\_启动窗口”添加 1 个列表框组件和 1 个按钮组件，列表框用来显示答案。

经过简单的分析可知：母鸡的数量不可能超过 19 只，公鸡的数量不可能超过 33 只，小鸡不能超过 100 只，所以将循环次数定为 19、33，这样是为了优化方案，尽量减少循环次数。当母鸡和公鸡数量确定后，小鸡的数量就等于 100 减公鸡数量和母鸡数量（等于已经符合了 1 个方程），在循环的中间用“如果真 ( )”命令进行判断，如果同时符合第 2 个方程，则将当前枚举出的数量显示到列表框中。

第二步，双击按钮，在“\_按钮 1\_被单击”子程序中输入代码：

变量名	类型	静态	数组	备注
母鸡数量	整数型			
公鸡数量	整数型			
小鸡数量	整数型			

```
--> 计次循环首 (19, 母鸡数量)
    --> 计次循环首 (33, 公鸡数量)
        小鸡数量 = 100 - (母鸡数量 + 公鸡数量)
        如果真 (母鸡数量 * 5 + 公鸡数量 * 3 + 小鸡数量 / 3 = 100)
            列表框1.加入项目 (“母鸡数量:” + 到文本 (母鸡数量) + “公鸡数量:”
                + 到文本 (公鸡数量) + “小鸡数量:” + 到文本 (小鸡数量), )
        --- 计次循环尾 0
    --- 计次循环尾 0
```

运行程序，单击按钮，答案即被显示在列表框中，如图 4-8 所示。



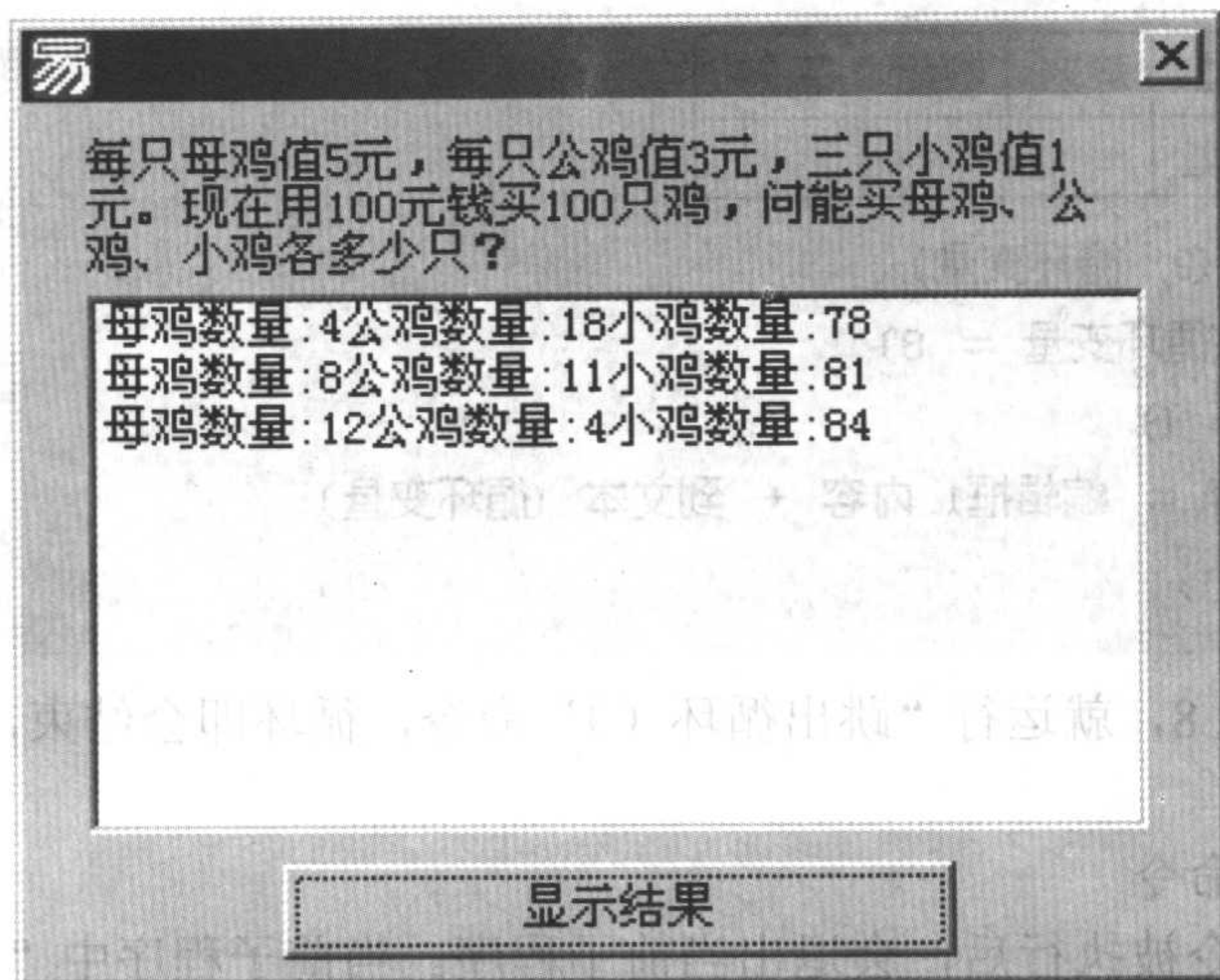


图 4-8 枚举结果

本例程参看随书光盘中的“枚举问题.e”。

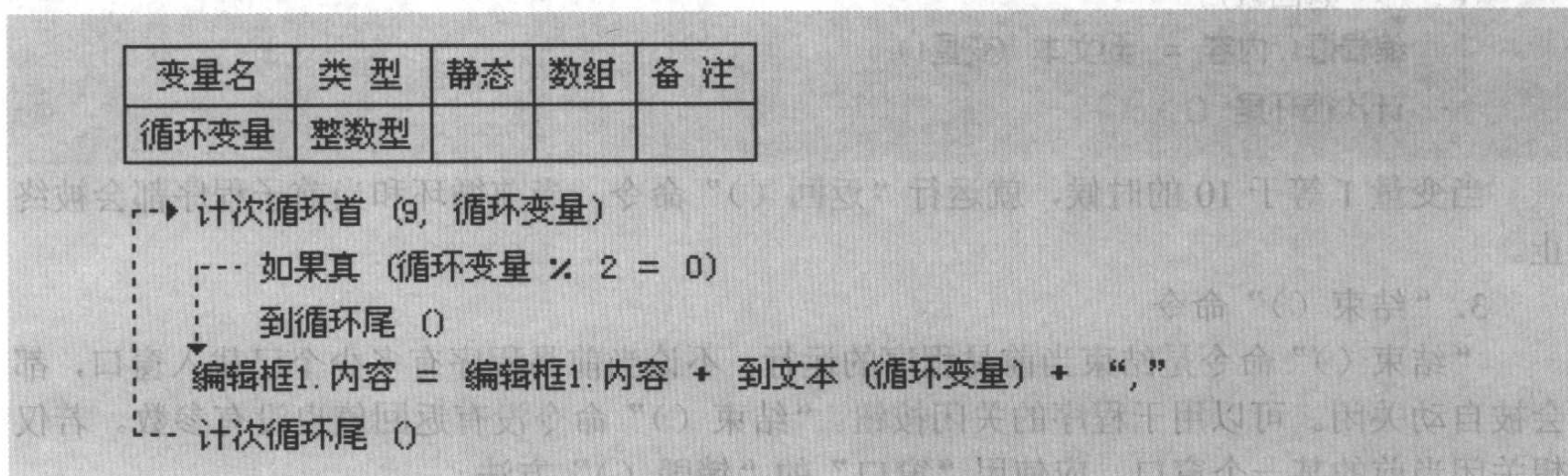
#### 4.2.4 跳转类流程控制命令

有了跳转类流程控制命令，可以更加方便的控制循环，“返回（）”和“结束（）”命令可以控制子程序和整个程序的结束。

##### 1. “到循环尾（）”和“跳出循环（）”命令

这两个跳转类流程控制命令都是用来控制循环的。当一个循环中运行了到“到循环尾（）”命令，将会直接跳到循环尾的代码处，当一个循环中运行了“跳出循环（）”命令，那么当前的循环就会结束，然后继续运行循环体以后的代码。例如：

(1) 让一个编辑框中只显示 10 以内的奇数“1, 3, 5, 7, 9”，代码如下：



当循环变量等于不想显示出的值，就会执行“到循环尾（）”命令，跳过显示到编辑框的代码。

(2) 如果在循环过程中，当某个条件产生，想提前结束该循环，可以使用“跳出循环（）”命令，如：





变量名	类型	静态	数组	备注
循环变量	整数型			

```
--> 计次循环首 (9, 循环变量)
-- 如果真 (循环变量 = 8)
-- 跳出循环 ()
-- 编辑框1.内容 = 编辑框1.内容 + 到文本 (循环变量)
-- 计次循环尾 ()
```

当循环变量等于 8，就运行“跳出循环 ()”命令，循环即会结束，所以编辑框会显示“1234567”

## 2. “返回 ()”命令

“返回 ()”命令被执行后，会退出当前子程序。当前子程序中“返回 ()”命令之后的代码将不再被执行，程序将自动跳转到调用本子程序语句的下一条语句处继续执行。

由于“返回 ()”命令后面的代码不被运行，所以“返回 ()”命令也经常被用于程序的侦错，在后面的章节会有介绍。有返回值的子程序必须使用“返回 ()”命令来返回执行结果，要注意实际返回值的数据类型要和子程序定义的返回值数据类型相匹配。例如返回日期时间型的数据：

返回 ([2004 年 11 月 15 日])

在循环中使用“返回 ()”命令，也可以终止一个循环的运行。例如：

变量名	类型	静态	数组	备注
变量1	整数型			

```
--> 计次循环首 (20, 变量1)
-- 如果真 (变量1 = 10)
-- 返回 ()
-- 编辑框1.内容 = 到文本 (变量1)
-- 计次循环尾 ()
```

当变量 1 等于 10 的时候，就运行“返回 ()”命令，当前循环和当前子程序都会被终止。

## 3. “结束 ()”命令

“结束 ()”命令是结束当前易程序的运行。不论当前易程序有多少个已载入窗口，都会被自动关闭。可以用于程序的关闭按钮。“结束 ()”命令没有返回值也没有参数。若仅想关闭当前的某一个窗口，应使用“窗口”的“销毁 ()”方法。

例如，在按下“关闭按钮”后，就将结束程序：

子程序名	返回值类型	公开	备注
_关闭按钮_被单击			

结束 ()

注：建议尽量在程序中使用销毁所有已载入窗口的方法来结束程序。



### 4.3 其他常用命令

后面章节将陆续介绍和章节主题相关的命令。所以本节将介绍一些后续章节中未提及的常用命令，以做参考。

#### 4.3.1 文本操作类命令

在编写程序时免不了对大量的文本型的数据进行操作，文本操作类的命令比较全面，前面已经用到过“分割文本（）”命令，下面介绍其他的常用文本操作命令。

##### 1. “取文本长度（）”命令

获取指定文本的字节长度，半角数字和字符为 1 个字节的长度，汉字和全角标点符号为 2 个字节的长度，如：

取文本长度（编辑框 1. 内容）

可以取出编辑框中文本的长度。

##### 2. “取文本左边（）”、“取文本右边（）”和“取文本中间（）”命令

这 3 个命令可以取出一段文本中任意位置的文本。如：

取文本左边（编辑框 1. 内容， 4）

可以将编辑框中的前 4 个字符取出来。

##### 3. “寻找文本（）”和“倒找文本（）”命令

从当前文本的指定位置开始寻找指定的文本，并返回最先找到该文本的位置。“寻找文本”是从指定文本的首部开始寻找，“倒找文本”相反。例如：

寻找文本（编辑框 1. 内容，“：”， 1， 假）

代码运行后会返回找到的第一个“：”的位置。

##### 4. “文本替换（）”命令

该命令可以将指定文本的某一部分用其他的文本替换。例如：

编辑框 1. 内容 = 文本替换（编辑框 1. 内容， 4， 2，“xx”）

将“编辑框 1”内容中第 4 个位置开始的 2 个字符替换成“xx”，并将结果显示在原编辑框中。

文本操作的命令通常都相互配合使用，打开随书光盘中例程“文本替换.e”，程序运行效果如图 4-9 所示。在按下按钮后，将编辑框中的 IP 地址，替换成“192.168.0.255”，并用信息框显示替换后的文本。



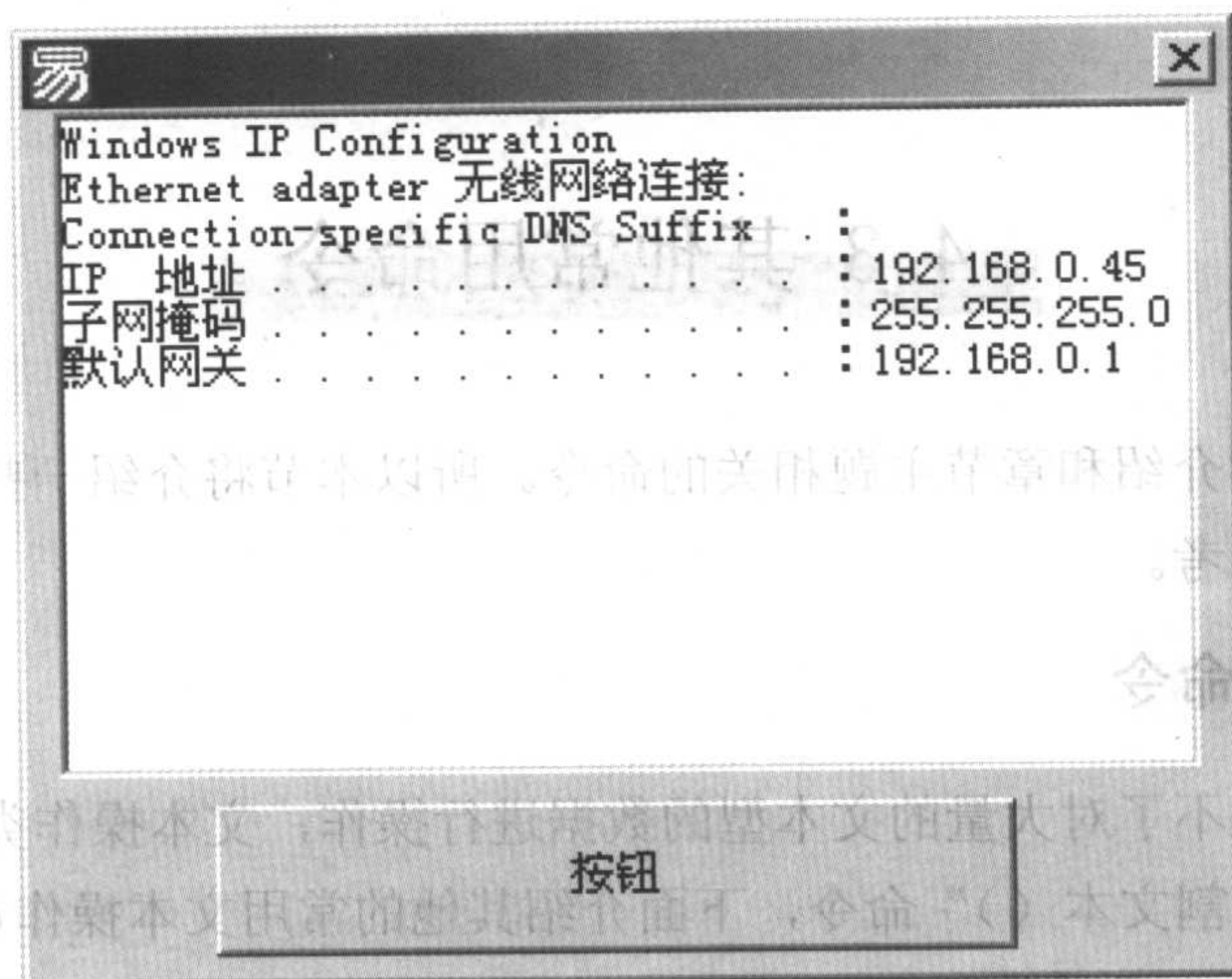


图 4-9 编辑框中显示的文本

在替换前首先要考虑将欲替换的文本的位置找到。IP 地址是容易改变，所以不能以数字串“192.168.0.45”作为查找对象。IP 地址前有一个“:”，可以先找到这个“:”，然后加上“:”占用的 2 字符，就可以得到 IP 地址文本的第一个字符的位置，找到了这个位置，就可以确定被替换文本的位置。但 IP 地址前的“:”不是第一个“:”，所以要连续寻找 2 次，第二次从第一个“:”的下一个位置开始寻找，就可以找到正确的位置了。

要将替换后的文本取出来，只要得到替换后的文本长度，就可以得到。可以先查找“子网掩码”的“子”的位置，然后用“子”的位置减 IP 地址第一个字符的位置，就可以得到 IP 地址的文本长度。程序用到的代码为：

子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
文本位置	整数型			
文本位置2				
取出的文本长度				

文本位置 = 寻找文本 (编辑框1.内容, “:”, 1, 假)

文本位置 = 寻找文本 (编辑框1.内容, “:”, 文本位置 + 1, 假)

编辑框1.内容 = 文本替换 (编辑框1.内容, 文本位置 + 2, 13, “192.168.0.255”)

文本位置2 = 寻找文本 (编辑框1.内容, “子”, 1, 假)

取出的文本长度 = 文本位置2 - (文本位置 + 2)

信息框 (取文本中间 (编辑框1.内容, 文本位置 + 2, 取出的文本长度), 0, )

### 4.3.2 时间操作类命令

时间操作类命令也是较常用的一类命令，可以对日期时间型数据进行操作。

#### 1. “到时间 ()”和“时间到文本 ()”命令



这2个命令用来在日期时间型数据和文本型数据之间转换。例如：

到时间 (“2004/2/2 12:30:25”)

“时间到文本”命令可以将指定的部分的时间转换成文本。例如：

时间到文本 (取现行时间 (), #日期部分)

## 2. “取现行时间 ()”命令

可以将当前系统的日期及时间取出，例如用编辑框显示当前的时间：

编辑框1. 内容 = 到文本 (取现行时间 ())

显示出的时间文本是一个包括“年月日时分秒”的完整时间，如果想取出时间中的指定部分，需要使用其他时间操作类型命令来实现。

## 3. 取指定时间部分的命令

用来取出指定时间部分的命令有：“取时间部分 ()”、“取年份 ()”、“取月份 ()”、“取星期几 ()”、“取小时 ()”、“取分钟 ()”、“取日 ()”、“取秒 ()”、“取日期 ()”、“取时间 ()”。其中“取日期 ()”和“取时间 ()”命令返回值是日期时间型，其他返回值都为整数型。

“取时间部分 ()”命令可以取出日期时间型数据中的指定部分数值。例如：

(1) 取当前时间的年份：

取时间部分 (取现行时间 (), #年份)

(2) 用编辑框显示当前时间的“年、月、日”：

编辑框1. 内容 = 到文本 (取日期 (取现行时间 ()))

### 4.3.3 位运算命令

位运算是指对数据进行二进制的逐位运算。计算机内部是采用二进制方式存储和处理数据的，输入到计算机的数字、字母、汉字等信息都以二进制的形式存储。

所谓二进制，就是以“逢二进一，借一当二”为原则，对数值进行计数的进位制，和我们日常使用的十进制类似，只不过十进制是“逢十进一”。

位的英文是 Bit，所以也常被称为比特位。

在计算机内部运算中常用的进位制有4种：

二进制：逢2进1，由数字0和1组成，以下标2或后缀B表示。

八进制：逢8进1，由数字0至7组成，以下标8或后缀Q表示。

十进制：逢10进1，由数字0至9组成，以下标10后缀D表示，该后缀可以省略。

十六进制：逢16进1，由数字0至9和字母A至F组成，以下标16或后缀H表示。

例如：2进制数1001010表示为1001010 (B)、八进制数234512表示为234512 (Q)、十六进制数4523ADF表示为4523ADF (H)，十进制数的后缀可以省略。

用不同进位制表示的数之间，可以根据一定的规则相互转换。

#### 1. 十六进制数、八进制数与二进制数之间的转换

一位十六进制数用四位二进制数表示，一位八进制数用3位二进制数表示。

二进制数转换为十六进制数时，以小数点位置为界，向两侧每四位分组，当两侧不足四位时补0。例如：101010.010101 (B) = 0010 1010.0101 0100 (B) = 2A.54 (H)

二进制数转换为八进制数时，以小数点位置为界，向两侧每三位分组，当两侧不足三位时补0。例如：101010.010101 (B) = 101, 010.010, 101 (B) = 52.25 (Q)



十六进制数转换为二进制数时，以小数点为界，每一位十六进制数转换为四位二进制数向两侧排列；八进制数转换为二进制数时，以小数点为界，每一位八进制数转换为三位二进制数向两侧排列。

## 2. 十进制数和二进制数之间的转换

### (1) 二进制数转换为十进制数

二进制数各位乘以与其对应的“权”之和即为该二进制数对应的十进制数。“权”是指以  $2^n$  代表二进制数的位，例如：

$$1011100.10111\text{B} = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} = 92.71875 (\text{D})$$

### (2) 十进制数转换为二进制数

十进制数转换为其他进制数时，整数部分和小数部分分别进行转换。整数部分采用除基取余法，小数部分采用乘基取整法。

使用除基取余法转换整数的方法是：

(1) 如果十进制整数小于要转换成的进位制的基数，则此十进制整数就是要转换成的进位制表示的数。

例如，将十进制整数 6 转换成八进制数，由于  $6 < 8$ ，十进制整数 6 也可以看做八进制数 6，即  $6 (\text{D}) = 6 (\text{Q})$ 。

(2) 如果十进制整数大于要转换成的进位制的基数，用该十进制整数除要转换成的进位制的基数，取余数，该余数就是转换后的低位数。此时，如果所得的商小于基数，则商就是转换后的高位数。

例如，十进制数 12 转换成八进制数， $12/8$ ，商为 1，余数为 4，由于  $4 < 8$ ，不再继续转换，则  $12 (\text{D}) = 14 (\text{Q})$ 。

(3) 如果上一步所得的商仍然大于要转换成的进位制的基数，继续第二步的操作，直到商小于基数为止。所取得的余数和商按下列顺序从左到右排列：

商、最后得到的余数、...、第二个得到的余数、第一个得到的余数。

例如，将十进制数 205 转换成二进制数，过程如下：

$205/2$ ，商为 102，第一个余数 1；

$102/2$ ，商为 51，第二个余数为 0；

$51/2$ ，商为 25，第三个余数为 1；

$25/2$ ，商为 12，第四个余数为 1；

$12/2$ ，商为 6，第五个余数为 0；

$6/2$ ，商为 3，第六个余数为 0；

$3/2$ ，商为 1，第七个余数为 1；

此时，商  $1 < \text{基数 } 2$ ，转换完毕，则得到的二进制数为：

1100110（商、第七个余数、第六个余数、第五个余数、第四个余数、第三个余数、第二个余数、第一个余数）。

使用乘基取整法转换小数的方法是：

对于被转换的十进制数的小数部分则应不断的乘以 2，并记下其整数部分，直到结果的小数部分为 0 为止。



例如：将十进制数 0.6875 转换为二进制数，过程如下：

$0.6875 \times 2 = 1.375$ ，小数点后第一位为 1；

$0.375 \times 2 = 0.75$ ，小数点后第二位为 0；

$0.75 \times 2 = 1.5$ ，小数点后第三位为 1；

$0.5 \times 2 = 1.0$ ，小数点后第四位为 1。

由于最后一步所得的乘积的小数部分为 0，转换结束，结果为 0.1011。

### 3. 易语言中的位运算命令

#### (1) “位取反 ( )” 命令

“位取反 ( )” 命令对二进制数值每一比特位的值取反，即 0 变为 1，1 变为 0，返回值是转换后的十进制数。

例如：

编辑框 1. 内容 = 到文本 (位取反 (80))

代码运行后，编辑框会显示“位取反”运算结果“-81”。

#### (2) “位与 ( )” 命令

“位与 ( )” 命令对二进制数值的共同比特位进行“与”运算，即如两个或多个数值的共同位均为 1，则返回值的对应位也为 1，否则为 0，运算完毕后，返回值是转换后的十进制数。

比如：

一个二进制数的第 4 位为 1，另一个二进制数的第四位为 1，则返回值的第四位为 1；

一个二进制数的第 4 位为 0，另一个二进制数的第四位为 1，则返回值的第四位为 0；

一个二进制数的第 4 位为 1，另一个二进制数的第四位为 0，则返回值的第四位为 0；

一个二进制数的第 4 位为 0，另一个二进制数的第四位为 0，则返回值的第四位为 0；

例如：

编辑框 1. 内容 = 到文本 (位与 (56, 89))

运行后可以得出的结果为“24”。56 和 89 分别转换成二进制数为：0011 1000 和 0101 1001，进行位与运算后即会得出结果“0001 1000”即“24”。

#### (3) “位或 ( )” 命令

“位或 ( )” 命令对二进制数值进行“或”运算，并将运算后结果以十进制返回。如两个或多个数值的共同位均为 0，则返回值的对应位也为 0，否则为 1。运算完毕后，返回值是转换后的十进制数。

一个数值的第 4 位为 1，另一个数值的第四位为 1，则返回值的第四位为 1；

一个数值的第 4 位为 0，另一个数值的第四位为 1，则返回值的第四位为 1；

一个数值的第 4 位为 1，另一个数值的第四位为 0，则返回值的第四位为 1；

一个数值的第 4 位为 0，另一个数值的第四位为 0，则返回值的第四位为 0；

例如：

编辑框 1. 内容 = 到文本 (位或 (56, 89))

运行后的结果为“121”。56 和 89 分别转换成二进制数为：0011 1000 和 0101 1001，进行或的运算后即会得出结果“0111 1001”即“121”。

#### (4) “位异或 ( )” 命令





“位异或 ( )”命令对二进制数值的共同比特位进行“异或”运算，并将运算结果以十进制返回。如果两个或多个数值的共同位相等（均为 0 或均为 1），则返回值的对应位就是 0，否则为 1。运算完毕后，返回值是转换后的十进制数。

比如：

- 一个数值的第 4 位为 0，另一个数值的第四位为 1，则返回值的第四位为 1；
- 一个数值的第 4 位为 1，另一个数值的第四位为 0，则返回值的第四位为 1；
- 一个数值的第 4 位为 1，另一个数值的第四位为 1，则返回值的第四位为 0；
- 一个数值的第 4 位为 0，另一个数值的第四位为 0，则返回值的第四位为 0；

例如：

编辑框 1.内容 = 到文本 (位异或 (56, 89))

运行后的结果为“97”。56 和 89 分别转换成二进制数为：0011 1000 和 0101 1001，进行异或的运算后即会得出结果“0110 0001”即“97”。

## 4.3.4 其他常用命令

1. “读入文件 ( )”和“写到文件 ( )”命令

“读入文件 ( )”命令将一个文件的所有数据读入程序，返回值是一个字节集型数据，在命令的参数中填入欲读入文件的全路径文件名，可以将读入的文件放在一个字节集型变量中，如：

变量 1 = 读入文件 ( “C:\Downloads\echs.zip” )

对读入的文件数据经过处理后，用“写到文件 ( )”命令写出至文件中。“写到文件 ( )”命令的第一个参数指定写出文件的全路径文件名，文件的扩展名要和文件格式相匹配，写出的文件才能正常访问。例如将字节集型数据“变量 1”中的内容写到文件中：

写到文件 ( “C:\echs.zip” , 变量 1 )

2. “寻找文件 ( )”命令

“寻找文件 ( )”命令可以在指定路径下寻找文件或目录，找到后就返回与条件匹配的文件名或目录名，如果没找到就返回一个空文本。命令的第一个参数为欲寻找的文件名，第二个参数为欲寻找文件的文件属性。例如寻找一个子目录：

寻找文件 ( “c:\aa” , #子目录 )

3. “创建目录 ( )”和“删除目录 ( )”命令

“创建目录 ( )”命令可以创建一个新目录，创建成功返回真，失败返回假。“删除目录 ( )”命令可以用来删除一个目录，删除成功返回真，失败返回假。

“创建目录 ( )”命令创建目录时，其父目录必须存在，否则会创建失败，即该命令不能一次性创建多级目录。因此，如果要创建多级子目录，需要判断每一级目录的父目录是否存在，一层一层的创建各级子目录。

下面就编写一个创建多级目录的程序。

首先新建一个易程序，在“\_启动窗口”中添加一个编辑框组件和一个按钮组件。编辑框组件用来输入欲创建的多级目录名。

双击按钮，在“\_按钮 1\_被单击”子程序中输入代码：



子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
分割后文本	文本型		0	
欲创建子目录	文本型			
是否创建成功	逻辑型			
变量1	整数型			

分割后文本 = 分割文本 (编辑框1.内容, "\", )

欲创建子目录 = 分割后文本 [1]

---▶ 计次循环首 (取数组成员数 (分割后文本) - 1, 变量1)

欲创建子目录 = 欲创建子目录 + "\" + 分割后文本 [变量1 + 1]

--- 如果真 (寻找文件 (欲创建子目录, #子目录) = "")

--- 如果 (创建目录 (欲创建子目录) = 真)

--- 是否创建成功 = 真

---▶ 是否创建成功 = 假

--- 跳出循环 ()

--- 计次循环尾 ()

信息框 (选择 (是否创建成功, "创建成功", "创建失败"), 0, )

首先使用了“分割文本 ()”命令，将编辑框中的文本以“\”为标志分割，分割后的文本就是各级子目录的目录名。使用“计次循环首 ()”命令，有几级子目录，就循环几次，由于子目录中包括一个驱动器盘符，所以循环次数要减 1。用“寻找文件 ()”命令寻找准备创建的目录，不存在即创建。用一个逻辑变量来记录是否创建成功。使用信息框显示创建的结果。如图 4-10 所示。

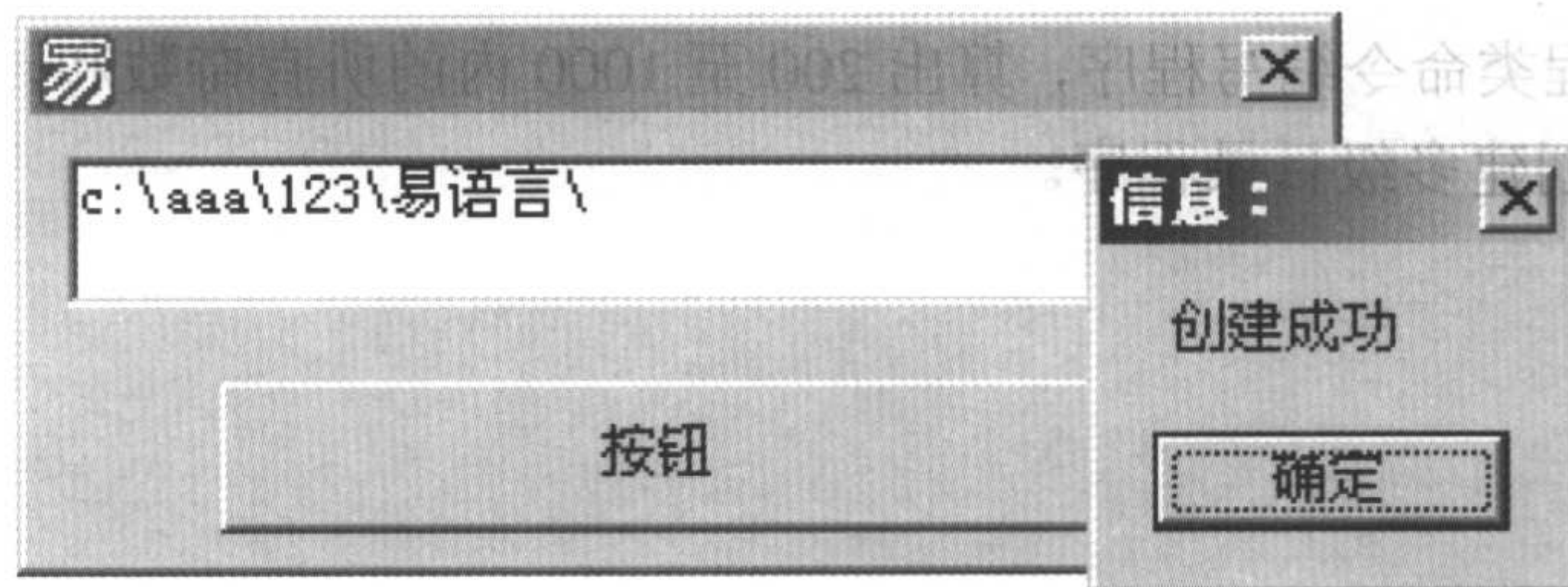


图 4-10 创建多级目录

例程参见随书光盘中的例程“创建多级目录.e”

#### 4. “打开文件 ()”命令

“打开文件 ()”用来打开一个指定的文件，成功返回被打开文件的文件号。虽然该命令只用来打开一个文件，并不对文件进行其他操作，但本命令取得的文件号，是很多文件操作类命令都要使用到的，如“读入文本 ()”和“写出文本 ()”命令、“读入数据 ()”





和“写出数据 ()”命令等等。使用此命令可以比读入文件命令实现更多的操作。

例如将一个文本文件打开并用“读入文本 ()”命令将读入的文本显示在编辑框中:

编辑框 1. 内容 = 读入文本 (打开文件 (“c:\帮助.txt”, , ), )

其他常用命令这里就不一一介绍了,大家可以根据命令的帮助系统来自学各命令的用法,并且在编程过程中逐步熟悉这些命令。

## 4.4 本章小结

易语言的集成化编辑环境提供了丰富的编程工具,如输入错误命令了,会即时进行编译,并且给出错误提示。大家也可以使用查找功能,快速找到您需要编辑的命令代码。左侧的支持库命令树可以方便您查找某命令的信息。命令输入后格式会自动规范,以便其他人查看源代码。

易语言的帮助功能十分强大,大家在学习本章时首先要学会自己看即时帮助,使用 F1 键查看相关的信息帮助可以起到独立自学的作用。

易语言提供**参数引导器**:当编写命令时,可以在命令上使用键盘右键展开命令的参数,以便于用户书写程序代码。减少程序代码输入中的错误。

《易语言知识库》提供了各命令的大量图解与例程,大家可以自行查看。

大家还可以进行以下练习:

1. “到文本 ()”命令的返回值是\_\_\_\_\_类型。
2. 分支类流程控制命令包括:\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
3. 循环类流程控制命令包括:\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
4. 跳转类流程控制命令包括:\_\_\_\_\_、\_\_\_\_\_、\_\_\_\_\_。
5. 用循环流程类命令编写程序,算出 200 至 1000 内的所有奇数和,看看有几种方法。
6. 自行编写创建多级目录程序。



## 第五章 子程序的编写与调用

子程序用来对一系列命令进行封装，实现模块化、重用及抽象，有利于程序的结构化开发，使程序结构更清晰。合理划分代码结构是软件成功的基本保障。

前面章节已经接触到了一些子程序，如“\_按钮 1\_被单击”子程序。通过对本章的学习，可以加深对子程序的认识，并能熟练的编写及调用子程序。

### 5.1 子程序的初步应用

#### 5.1.1 子程序的分类

易语言中的子程序可以分为两大类：“事件子程序”和“用户自定义子程序”。

前面曾见过的“\_按钮 1\_被单击”子程序，是在按钮被单击事件产生时程序需要执行的动作。这种对应组件所发生事件子程序，称作组件的“事件子程序”。例如：按钮有被单击、被双击、左键被按下等事件，选中按钮后，在属性面板中的事件列表中选择这些事件，就可以生成对应的事件子程序，大家可以在这些事件子程序中写入具体执行的代码。运行时，一旦这些事件产生，就会自动执行相应的子程序。事件子程序的名称、返回值类型和参数个数都是系统定义的，不允许用户任意修改。

“用户自定义子程序”是由用户创建，其参数个数和返回值都由用户自行定义子程序。用户可以根据需要在程序设计时对其任意修改。

使用子程序的好处很多，譬如有段代码在程序中多处被重复调用，此时就可以将其编写到一个子程序中，不仅减少了代码输入的重复劳动，而且需要修改这段代码时，只要修改一个地方即可，而不用在程序中逐个修改，即实现重用；再譬如要实现一个相对复杂的功能，如果全部代码写在一起，发现错误就无法确认问题来源，此时可将问题分解为多个简单问题，使用子程序逐个实现，这样有利于提高代码的正确性和清晰度。

组件的事件子程序在书中各章均有提及，且格式相对固定，因此本章将重点介绍自定义子程序的定义与使用。

#### 5.1.2 用户自定义子程序的创建

子程序是存在于程序集中的，由程序集分组管理。若一个新建的易程序还没有进行任何操作是没有程序集的，这时可以使用菜单：“插入”→“程序集”来新建一个程序集放置自己所写的用户自定义子程序；或通过窗体空白处双击鼠标，会自动创建该窗口对应的窗口程序集，并自动生成“\_启动窗口\_创建完毕”子程序（创建完毕事件子程序可包含本





窗口创建时执行的程序代码);或双击按钮等组件,也会创建该窗口对应的窗口程序集及组件对应缺省事件子程序。此时就可以在当前窗口程序集中添加新的子程序了。如图 5-1 所示。

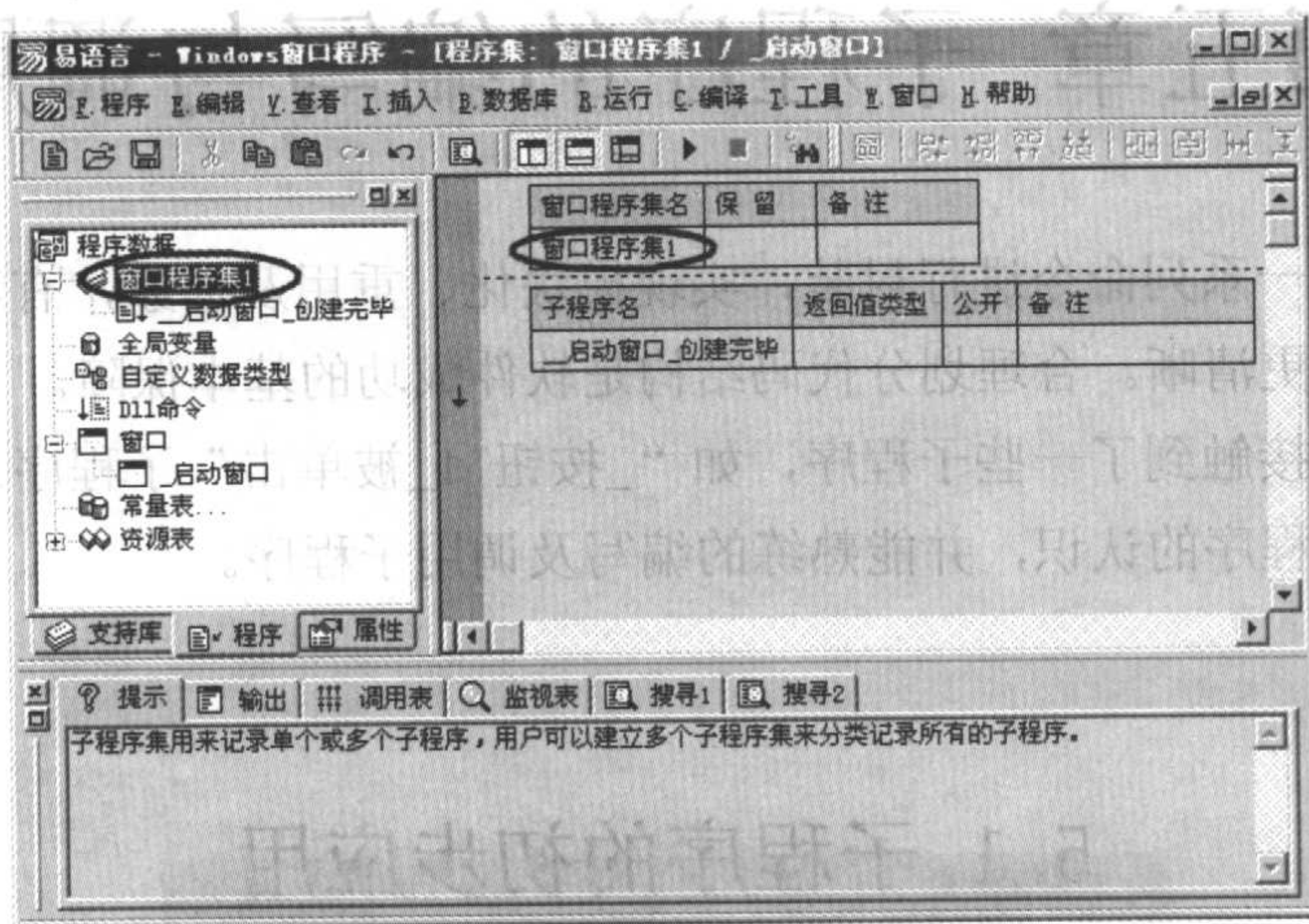


图 5-1 创建窗口程序集

用户自定义子程序可以放在窗口程序集中,或自定义的程序集中,当然,本书建议您将当前窗口用到的用户自定义子程序放在当前窗口程序集中,这样就可以非常方便地引用程序集变量和窗口中的组件,而且查找起来也较为方便。

2. 若程序集已建立,可通过程序面板切换到程序集中操作,或通过窗口菜单切换。

定义一个新的子程序可以通过在程序编辑界面按下“Ctrl+N”键;或在程序编辑界面点击鼠标右键,在弹出菜单中选择第一项“新子程序”;或者选择易语言菜单“插入”→“新子程序”来新建子程序。如图 5-2 所示。

创建后的子程序可以根据需要修改其名称,建议在定义子程序名称时,尽量选择能体现其功能的名称,如“求和”、“统计字符”等。

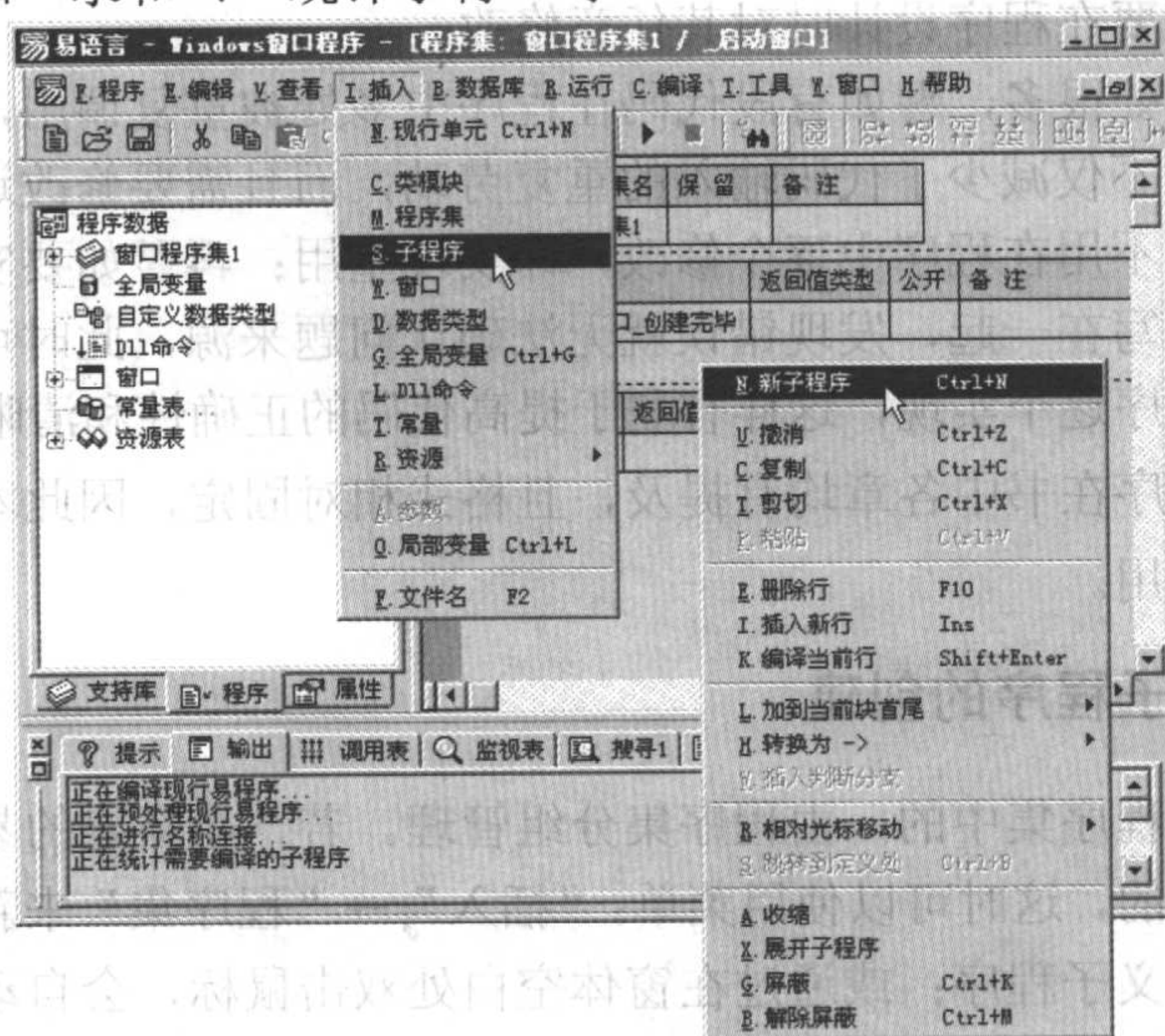


图 5-2 插入新子程序



### 5.1.3 子程序的调用

子程序的调用方法和命令的调用方法相同，输入子程序的名称就可以调用该子程序。如果子程序要求提供参数，在括号中要按照子程序参数定义的数据类型和参数个数顺序填写参数；如果需要用变量保存子程序返回值，必须使用和子程序返回值相同的数据类型变量。

### 5.1.4 返回值和参数的定义

使用“返回（）”命令，可以将子程序处理后的结果返回给调用它的程序使用。

为子程序定义返回值，需要在子程序的“返回值类型”单元格中定义其返回值的数据类型，同时，在子程序任意一个结束分支中，都必须使用“返回（）”命令将欲返回的值回传。

为子程序增加参数定义，首先在子程序名称上按回车键增加一个空白参数定义，修改该参数的名称和数据类型即可。参数的“类型”如果为空，系统默认为整数型参数。如果某项参数的“可空”类型未被设置，在调用该子程序时必须为此参数提供初始值。

子程序的参数在当前子程序内可以当作变量引用。例如：

子程序名	返回值类型	公开	备注		
相加运算	整数型				
参数名	类型	参考	可空	数组	备注
加数1	整数型				
加数2	整数型				

返回 (加数1 + 加数2)

上述子程序定义中，子程序名为“相加运算”，返回值类型为“整数型”，子程序有 2 个整数型参数，分别为“加数 1”和“加数 2”。

该子程序可以对参数做相加运算，并将运算结果回传。调用该子程序的方法如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

编辑框1.内容 = 到文本 (相加运算 (1345462, 33456730))

### 5.1.5 编写一个子程序

下面用一个实例讲解如何编写子程序。本例中该子程序用来统计在一段文本中某字符出现的个数。

首先新建一个易程序，在窗口中添加 2 个编辑框组件和 1 个按钮组件，一个编辑框用来输入被查找的文本，可将其“允许多行”属性设为“真”便于查看；另一个用来输入欲查找的文本。按钮组件的标题改为“开始查找”。如图 5-3 所示。



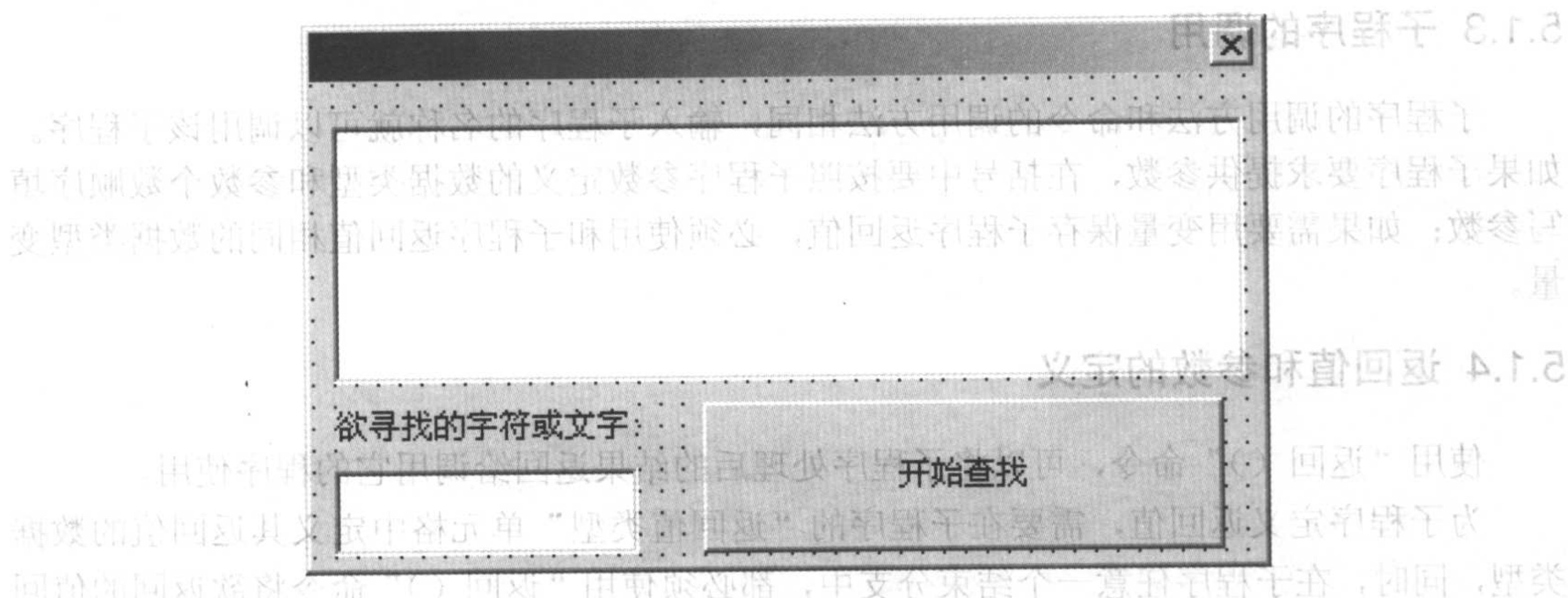


图 5-3 查找文本例程界面

进入窗口对应的程序集，新建一个子程序，将子程序返回值设置为整数型，并定义 2 个文本型参数，分别为“被寻找文本”和“欲寻找文本”，然后在子程序中输入如下代码：

子程序名	返回值类型	公开	备注		
查找文本个数	整数型				
参数名	类型	参考	可空	数组	备注
被查找文本	文本型				
欲查找文本	文本型				

变量名	类型	静态	数组	备注
欲查找文本长度	整数型			
找到位置	整数型			
记数	整数型			

```

欲查找文本长度 = 取文本长度 (欲查找文本)
找到位置 = 寻找文本 (被查找文本, 欲查找文本, 1, 假)
--> 判断循环首 (找到位置 ≠ -1)
    记数 = 记数 + 1
    找到位置 = 寻找文本 (被查找文本, 欲查找文本, 找到位置 + 欲查找文本长度, 假)
--- 判断循环尾 0
返回 (记数)
    
```

“查找文本个数”自定义子程序中使用了“寻找文本 ()”命令，该命令会返回指定文本在另一文本中，从指定位置开始最先出现的位置，如果没有找到返回-1。程序循环寻找指定文本，直到失败自动跳出循环。循环结束后，使用返回命令，将统计的文本个数回传。

在“\_按钮 1\_被单击”事件子程序中用信息框显示查找的结果，代码如下：



子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
找到个数	整数型			

找到个数 = 查找文本个数 (编辑框1.内容, 编辑框2.内容)

信息框 (“找到” + 到文本 (找到个数) + “个” + #左引号 + 编辑框2.内容 + #右引号, 0, )

试运行程序，在编辑框 1 中输入一段文本，然后在编辑框 2 中输入欲查找的文本，按下按钮，信息框显示查找到的文本个数。如图 5-4 所示。

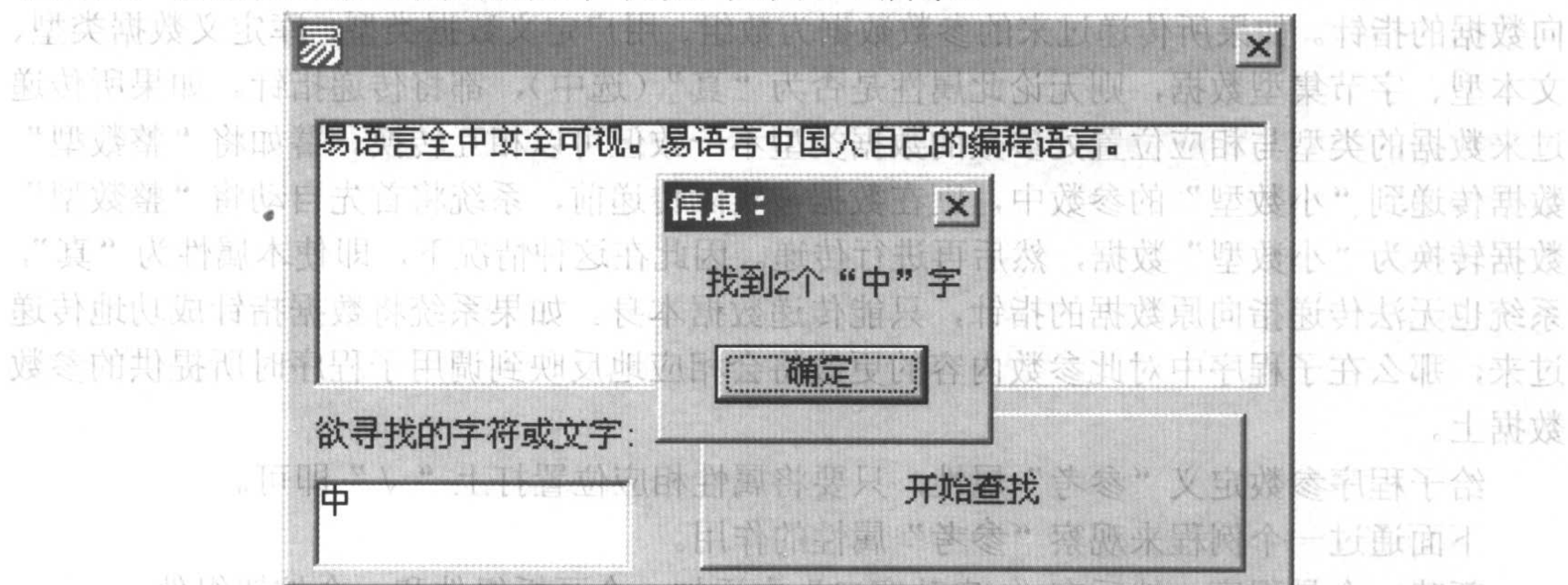


图 5-4 “寻找文本个数”例程运行结果

本例程参看随书光盘中“寻找文本个数.e”。

### 5.1.6 子程序指针

易语言中部分命令必须以子程序在内存中的起始地址作为参数，如“启动线程 ( )”命令；使用某些外部 API 函数时，也同样需要知道调用的子程序在内存中的起始地址。在易语言中，以子程序指针型变量记录子程序在内存中的起始地址。赋值方法为“&”+子程序名。如：

变量 1 = &子程序 1

启动线程 (变量 1, )

赋值之后，“变量 1”（类型为“子程序指针型”）存储的就是“子程序 1”的可执行代码首地址。





## 5.2 子程序的参数属性

子程序每个参数都存在“参考”、“可空”和“数组”三个属性。这三个属性决定了子程序参数的几种特性。本节将介绍它们的作用和用法。

### 5.2.1 参数的“参考”属性

子程序参数的“参考”属性用于设置系统为当前子程序参数传递数据时是否为传递指向数据的指针。如果所传递过来的参数数据为数组、用户定义数据类型、库定义数据类型、文本型、字节集型数据，则无论此属性是否为“真”（选中），都将传递指针。如果所传递过来数据的类型与相应位置处参数的数据类型不一致但可以相互转换。譬如将“整数型”数据传递到“小数型”的参数中，则在数据被实际传递前，系统将首先自动将“整数型”数据转换为“小数型”数据，然后再进行传递。因此在这种情况下，即使本属性为“真”，系统也无法传递指向原数据的指针，只能传递数据本身。如果系统将数据指针成功地传递过来，那么在子程序中对此参数内容的更改将会相应地反映到调用子程序时所提供的参数数据上。

给子程序参数定义“参考”属性，只要将属性相应位置打上“√”即可。

下面通过一个例程来观察“参考”属性的作用。

新建一个易程序，然后在“\_启动窗口”中添加一个画板组件和一个按钮组件。

双击按钮组件，在代码编辑界面新建一个子程序，定义子程序名为“测试”，为该子程序定义2个参数，均为整数型；参数名分别设为“参考参数”和“非参考参数”，将“参考参数”的“参考”属性上打上“√”：

子程序名	返回值类型	公开	备注		
测试					
参数名	类型	参考	可空	数组	备注
参考参数	整数型	√			
非参考参数	整数型				

参考参数 = 1000

非参考参数 = 1000

在“\_按钮1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注		
_按钮1_被单击					
变量名	类型	静态	数组	备注	
变量1	整数型				
变量2	整数型				

测试 (变量1, 变量2)

画板1.滚动写行 (“参考参数中的变量:” + 到文本 (变量1), “非参考属性中的变量:” + 到文本 (变量2))



运行程序，观察运行结果。如图 5-5 所示。

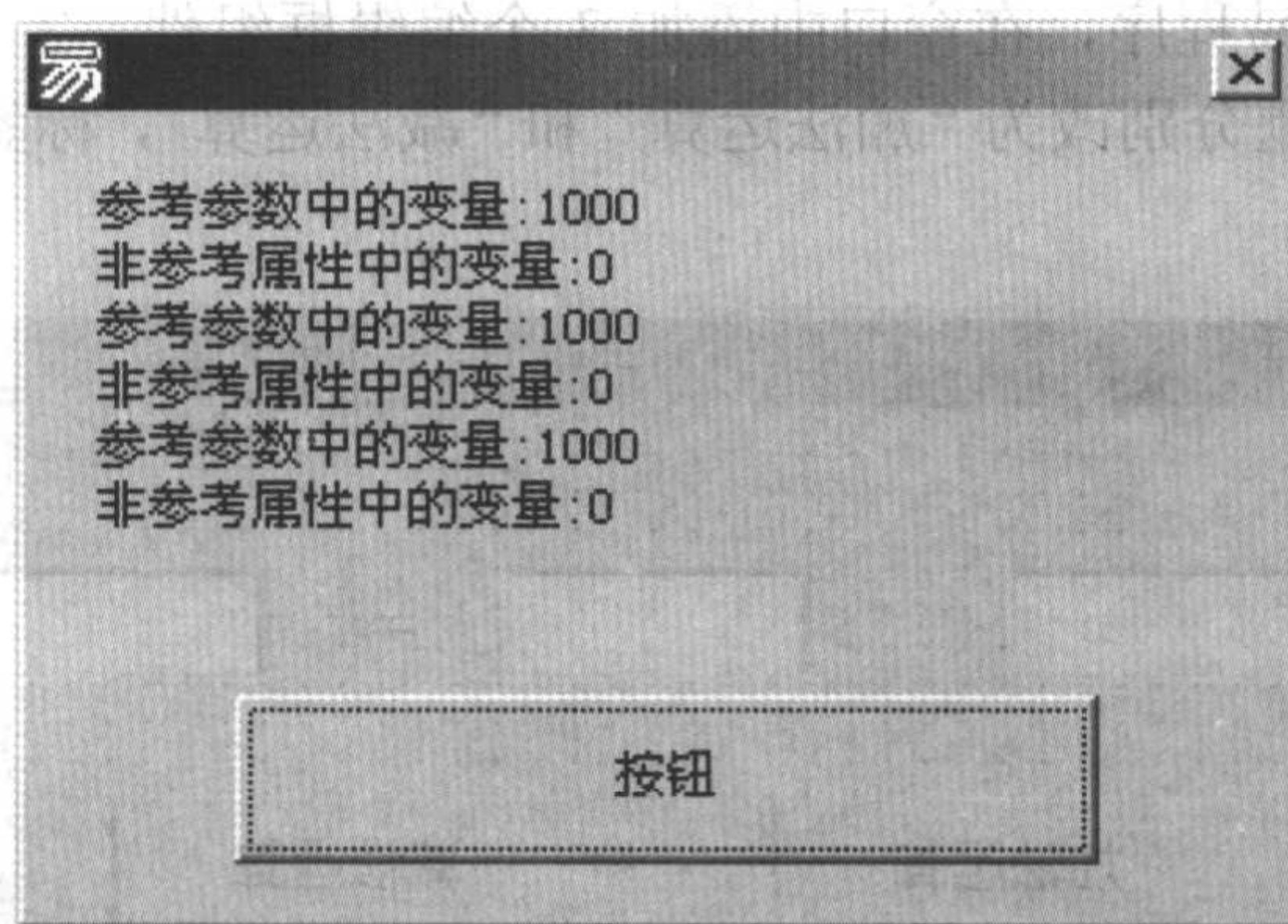


图 5-5 “参考”属性对变量的影响比较

上述程序中，每次点击按钮，就会调用“测试”子程序对 2 个参数进行修改。从画板的显示结果可以看出，对应“参考参数”的变量 1，因子程序内部对相应参数的操作而被修改，而对应“非参考参数”的变量 2 没有发生任何变化。

本例程参考随书光盘中的例程“参数的参考属性.e”。

### 5.2.2 参数的“可空”属性

易语言自带的支持库命令中有一些参数可以被省略，一般查找帮助时可以看到可省略的部分用中括号括起来了。如“时间到文本 ( )”命令的第二个参数是可以省略的，它也可有一个默认的数据，此命令帮助如下：

调用格式：〈文本型〉时间到文本（欲转换到文本的时间，[转换部分]）

将指定时间转换为文本并返回。

参数<1>的名称为“欲转换到文本的时间”，类型为“日期时间型 (date)”。

参数<2>的名称为“转换部分”，类型为“整数型 (int)”，可以被省略。参数值可以为以下常量：1、#全部转换；2、#日期部分；3、#时间部分。如果省略了本参数，默认为“#全部转换”。

用户自定义子程序也可以实现这样的功能。

在设计子程序的时候，有些参数可能经常用到相同的数据。如果每次调用该子程序，均需手工指定此数据作为参数初始值，那是一件非常繁琐的事情。在这种情况下，可以将此数据作为该参数的默认值，内置在子程序中，并将参数属性设为“可空”类型。当子程序被调用时，该参数没有指定具体的初始值，就为该参数赋予默认值参与运算。如果某项参数的“可空”类型未被设置，在调用该子程序时必须为此参数提供数据。

定义“可空”属性的方法和定义“参考”属性相同，将该属性打上“√”即可。如果需要检测当前子程序被调用时，该参数是否传递了数据，可用“是否为空 ( )”命令进行确认。

下面编写一个可以进行加法或减法运算的子程序，来了解“可空”属性的实际应用。

本例程将加减运算的功能制作成一个子程序，通过第三个参数确定是作加法计算还是



减法计算，如果第三个参数为空进行加法运算，不为空则进行减法运算。

第一步，新建一个易程序，在窗口中添加 3 个编辑框组件，2 个按钮组件和 1 个标签组件，将按钮组件的标题分别改为“加法运算”和“减法运算”，标签组件的标题改为“=”。如图 5-6 所示。

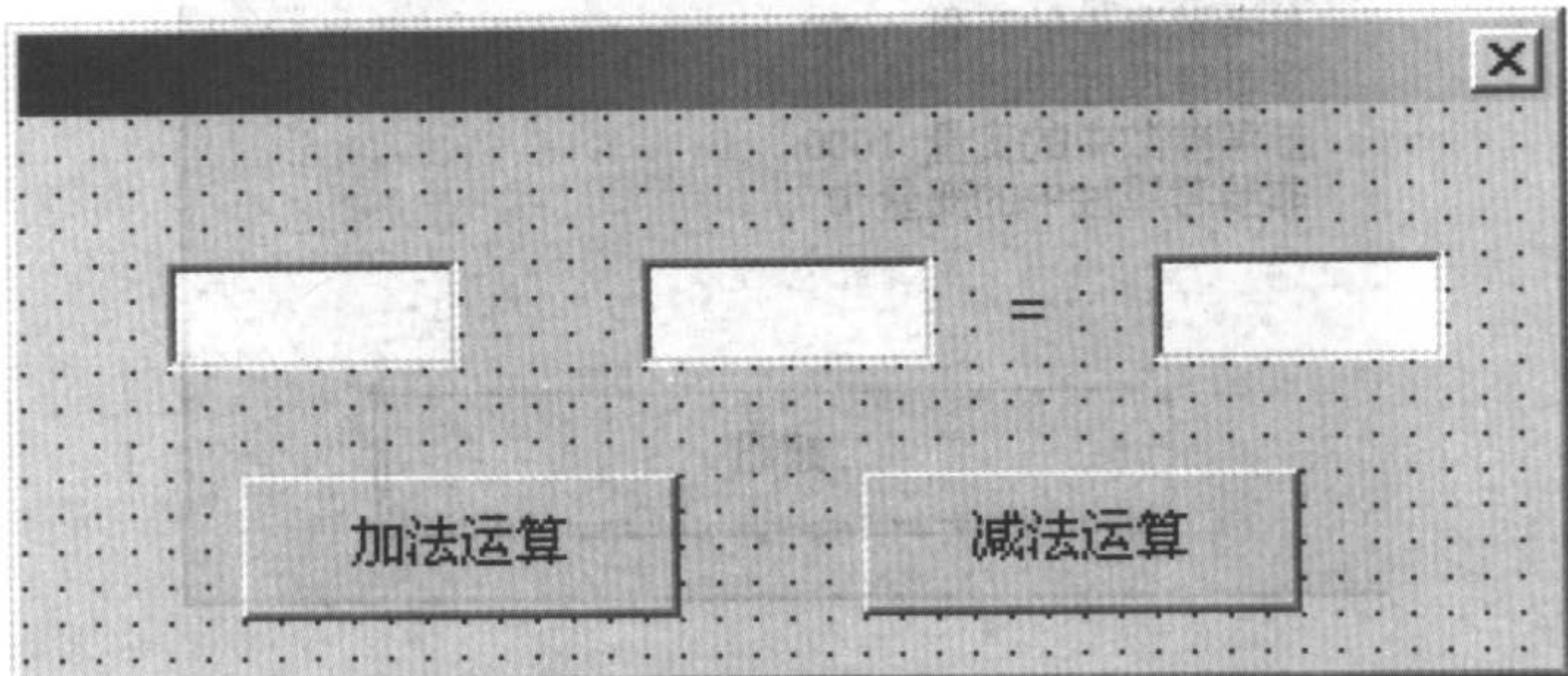


图 5-6 加减法运算例程界面

第二步，切换到代码编辑界面，新建一个子程序，将子程序名改为“加减运算”。给该子程序定义 3 个整数型参数，参数名分别为“被运算数”、“运算数”和“进行的运算”，将“进行的运算”参数的“可空”属性被设置，在子程序中输入如下程序代码：

子程序名	返回值类型	公开	备注		
加减运算	整数型				
参数名	类型	参考	可空	数组	备注
被运算数	整数型				
运算数	整数型				
进行的运算	整数型		✓		

```

如果 (是否为空 (进行的运算))
    返回 (被运算数 + 运算数)
返回 (被运算数 - 运算数)
    
```

子程序中，“进行的运算”参数控制子程序进行何种运算，该参数为空进行加法运算，该参数不为空则进行减法运算。

第三步，回到“\_启动窗口”双击“减法运算”按钮，然后在“\_按钮 1\_被单击”和“\_按钮 2\_被单击”子程序中分别输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

编辑框3.内容 = 到文本 (加减运算 (到数值 (编辑框1.内容), 到数值 (编辑框2.内容), ))

---

子程序名	返回值类型	公开	备注
_按钮2_被单击			

编辑框3.内容 = 到文本 (加减运算 (到数值 (编辑框1.内容), 到数值 (编辑框2.内容), 1))



可以看到，在“\_按钮 1\_被单击”子程序中，省略了“加减运算”子程序第 3 个参数的填写。

最后，运行程序，查看运行后的效果。如图 5-7 所示。

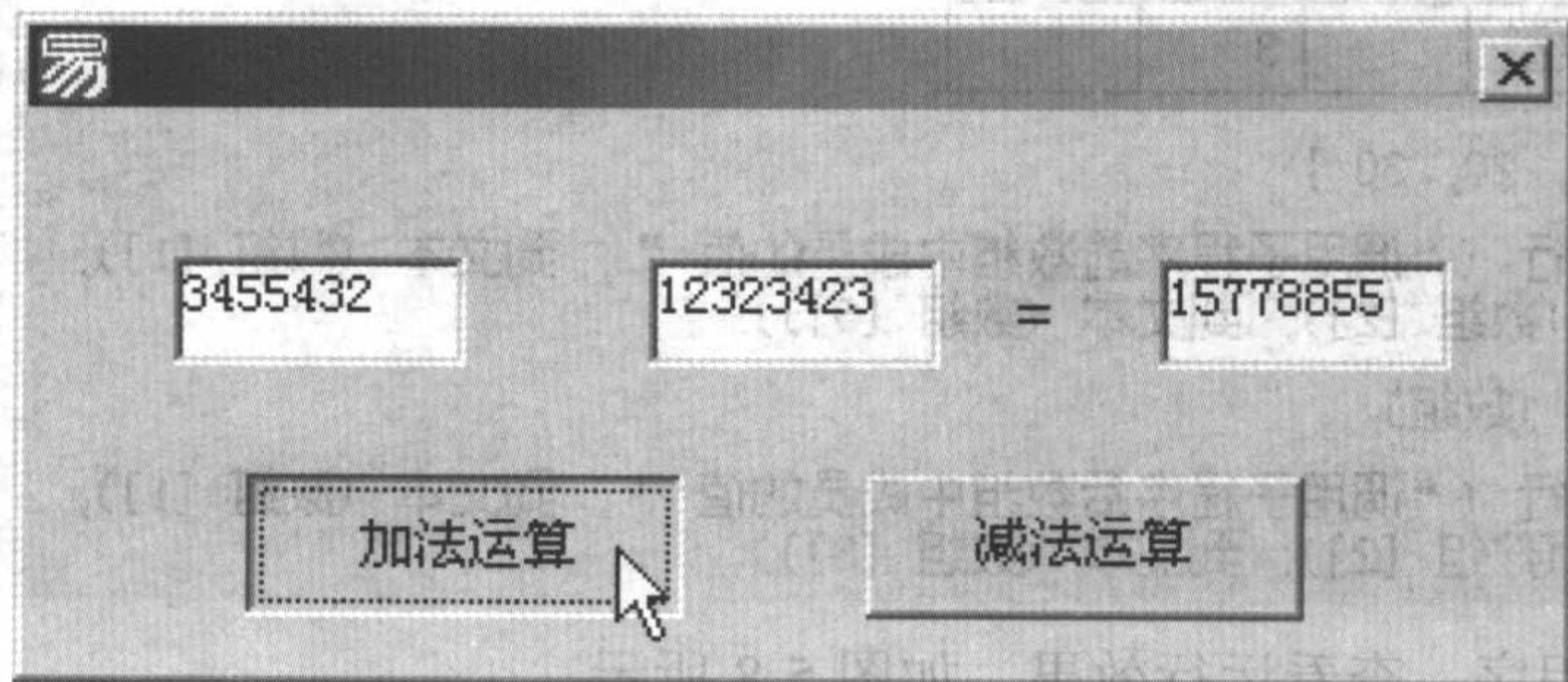


图 5-7 加减法运算例程的运行效果

程序运行时点击左边按钮，程序运行的“加减运算”子程序第三个参数没有填参数，因此进行的是加法计算。点击右边按钮，程序运行的“加减运算”子程序第三个参数设置参数为 1，因此进行的是减法计算。本例程可以参考随书光盘中的例程“加减运算.e”。

### 5.2.3 参数的“数组”属性

子程序参数也可以接收数组，如系统的“播放 MP3 (, )”命令的第二个参数就可以接收数组。而用户自定义子程序也可以实现这个功能，接收一个数组，在子程序内部进行处理。

子程序参数定义了“数组”属性后，就可以给这个参数中填写一个数组变量。

子程序的参数如果是“数组”型变量，就自动具有了“参考”属性。

定义参数的“数组”属性，在该参数的数组属性上打“√”。

下面就编写一个例程，来了解数组参数的作用。

第一步，新建一个易程序，在窗口中添加一个画板和一个按钮组件。

第二步，双击按钮组件，在程序设计界面新增一个子程序，然后将子程序名改为“测试数组属性”，给该子程序定义一个参数，将参数的名称改为“数组”，并将参数的数组属性选中：

子程序名	返回值类型	公开	备注		
测试数组属性					
参数名	类型	参考	可空	数组	备注
数组	整数型			✓	

数组 = { 100, 200, 300 }

第三步，在按钮 1 被单击的子程序中输入代码：





子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
数组	整数型		3	

数组 = { 10, 20, 30 }

画板1.滚动写行 (“调用子程序前数组中成员的值:”, 到文本 (数组 [1]), 到文本 (数组 [2]), 到文本 (数组 [3]))

测试数组属性 (数组)

画板1.滚动写行 (“调用子程序后数组中成员的值:”, 到文本 (数组 [1]), 到文本 (数组 [2]), 到文本 (数组 [3]))

最后试运行程序，查看运行效果，如图 5-8 所示。

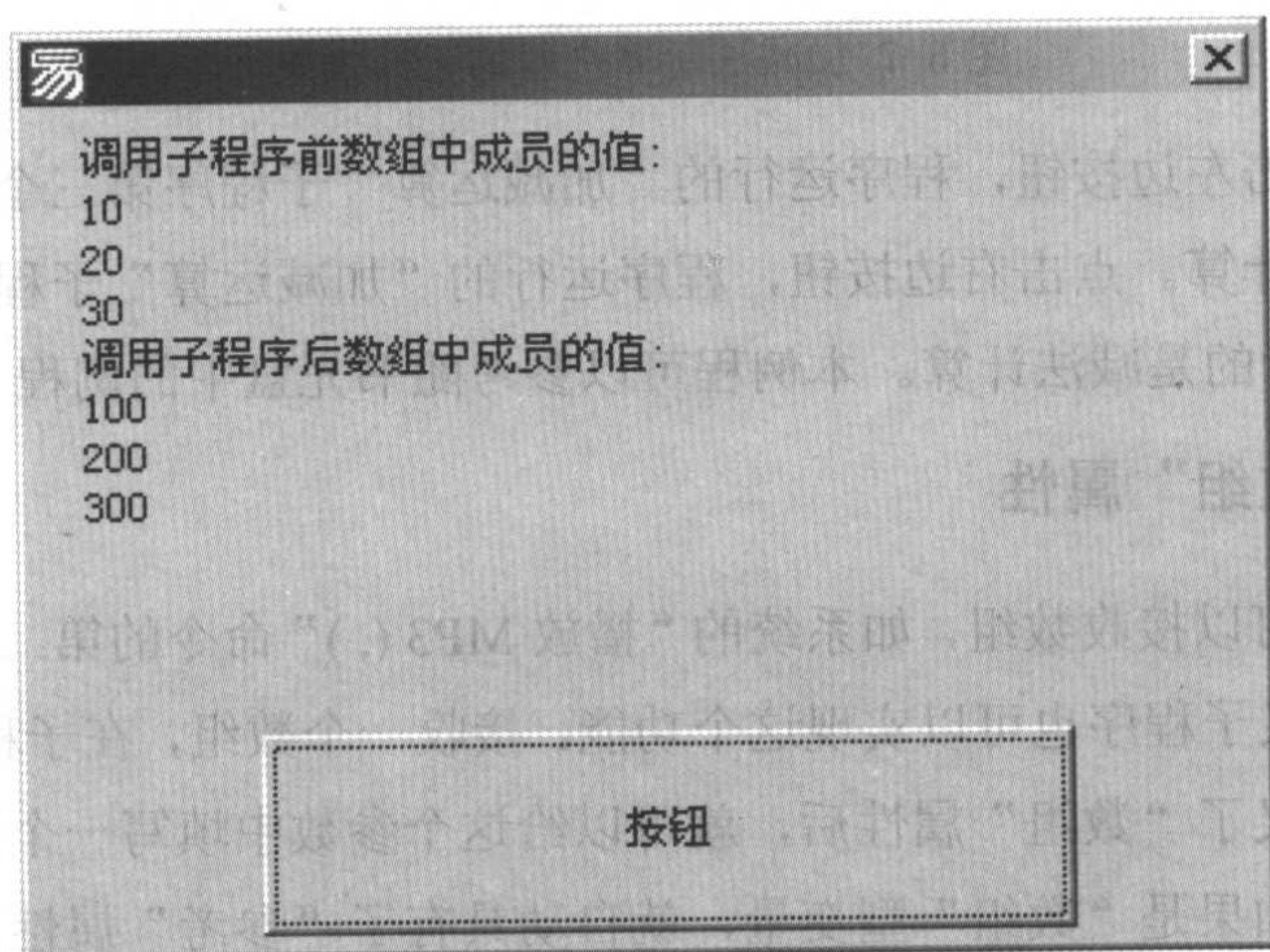


图 5-8 数组属性例程运行效果

运行后，画板显示出调用子程序前后数组变量中各成员的值，在调用子程序后，可以看出数组参数自动具有了参考属性，数组变量中成员的值发生了改变。

## 5.3 编写寻找文件子程序

大家可能经常使用 Windows 的搜索文件功能，可以在指定目录及其子目录中寻找特定条件的文件。现在来看该如何编写一个寻找文件的子程序，实现多级目录的文件搜索。

这是一个递归算法：子程序在指定的根目录中检索所有文件，对比文件名。然后检索根目录中的所有子目录，对每一个子目录调用自身子程序深入其中再次进行文件检索。不断的自身调用直到所有的子目录全部检索完毕。

第一步，新建一个易程序，按图5-9所示，在窗口中添加列表框组件与按钮组件各一个，编辑框组件与标签组件各两个。按钮与标签的标题按图中所示修改。



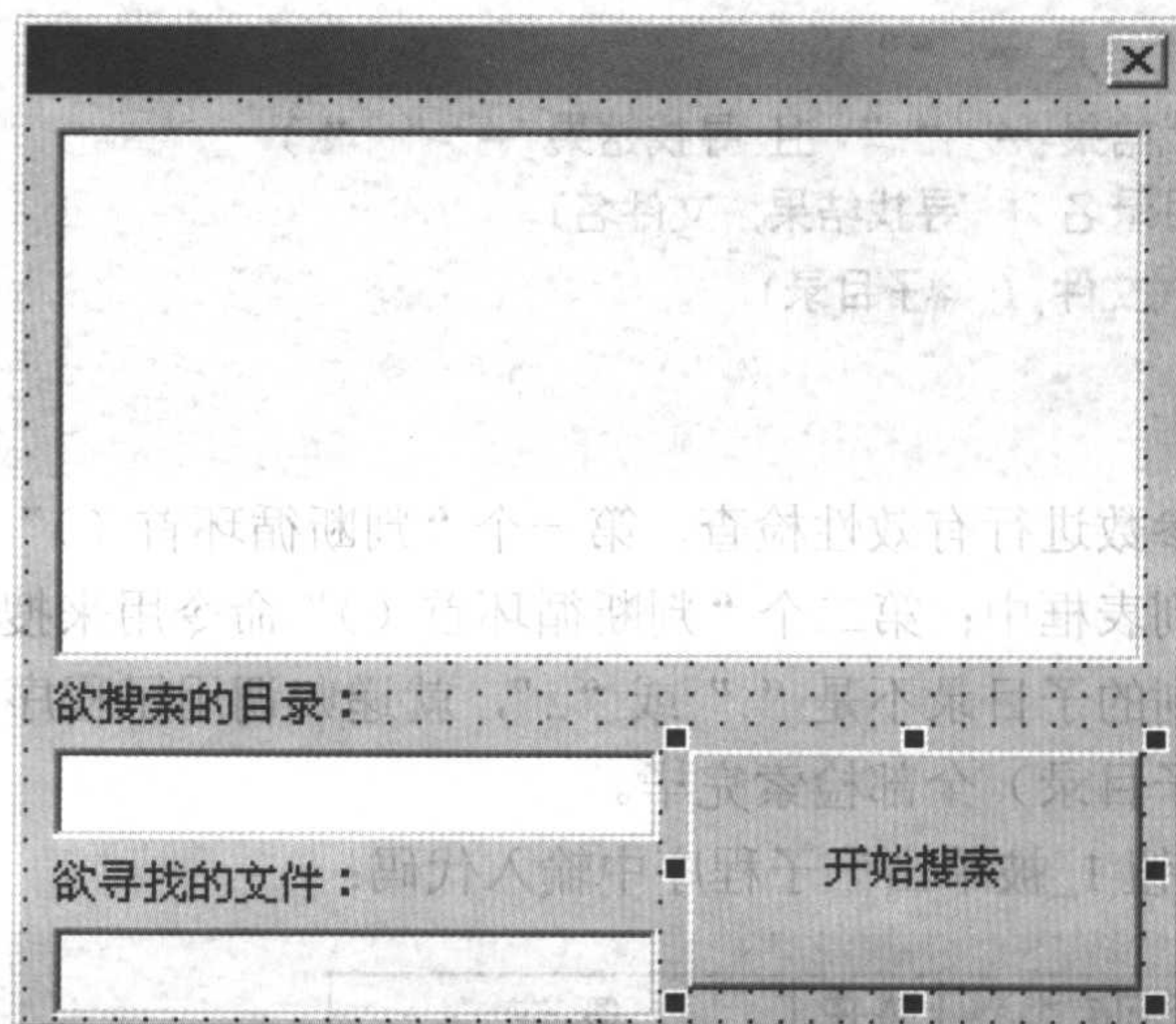


图 5-9 搜索文件例程界面

第二步，切换到代码设计界面，新建一个子程序，将子程序名改为“搜索文件”，给子程序定义 2 个文本型参数，分别为“目录名”和“文件名”，可以将“文件名”参数的“可空”属性选中，默认文件名参数为空的时候就寻找所有文件。在子程序中输入代码：

子程序名	返回值类型	公开	备注		
搜索文件					
参数名	类型	参考	可空	数组	备注
目录名	文本型				
文件名	文本型		✓		

变量名	类型	静态	数组	备注
寻找结果	文本型			

```

--- 如果真 (目录名 = "")
    信息框 ("错误, 搜索目录不能为空", #错误图标, )
    返回 ()

```

```

--▶ 如果真 (文件名 = "")
    文件名 = "*. *"

```

```

--▶ 如果真 (取文本右边 (目录名, 1) ≠ "\")
    目录名 = 目录名 + "\"

```

```

寻找结果 = 寻找文件 (目录名 + 文件名, )

```

```

--▶ 判断循环首 (寻找结果 ≠ "")
    列表框1. 加入项目 (目录名 + 寻找结果, )
    寻找结果 = 寻找文件 ( )

```

```

--- 判断循环尾 ()

```

```

寻找结果 = 寻找文件 (目录名 + "*. *", #子目录)

```





```
--> 判断循环首 (寻找结果 ≠ "")
-- 如果真 (寻找结果 ≠ "." 且 寻找结果 ≠ "..")
    搜索文件 (目录名 + 寻找结果, 文件名)
    寻找结果 = 寻找文件 ( #子目录)
-- 判断循环尾 ()
```

子程序中首先对参数进行有效性检查，第一个“判断循环首 ()”命令，将指定目录的所有匹配文件加入到列表框中；第二个“判断循环首 ()”命令用来搜索子目录，如果搜索到子目录，并且搜索到的子目录不是“.”或“..”，就递归调用子程序自身。直到所有子目录（包括子目录下的子目录）全部检索完毕。

第三步，在“\_按钮1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

列表框1. 清空 ()

搜索文件 (编辑框1. 内容, 编辑框2. 内容)

第四步，运行程序，搜索目录编辑框中输入“c:\”，寻找文件编辑框中输入“\*.mp3”，点击按钮后，列表框列举出 c 盘中所有搜索到的 Mp3 文件。运行效果如图 3-10 所示。

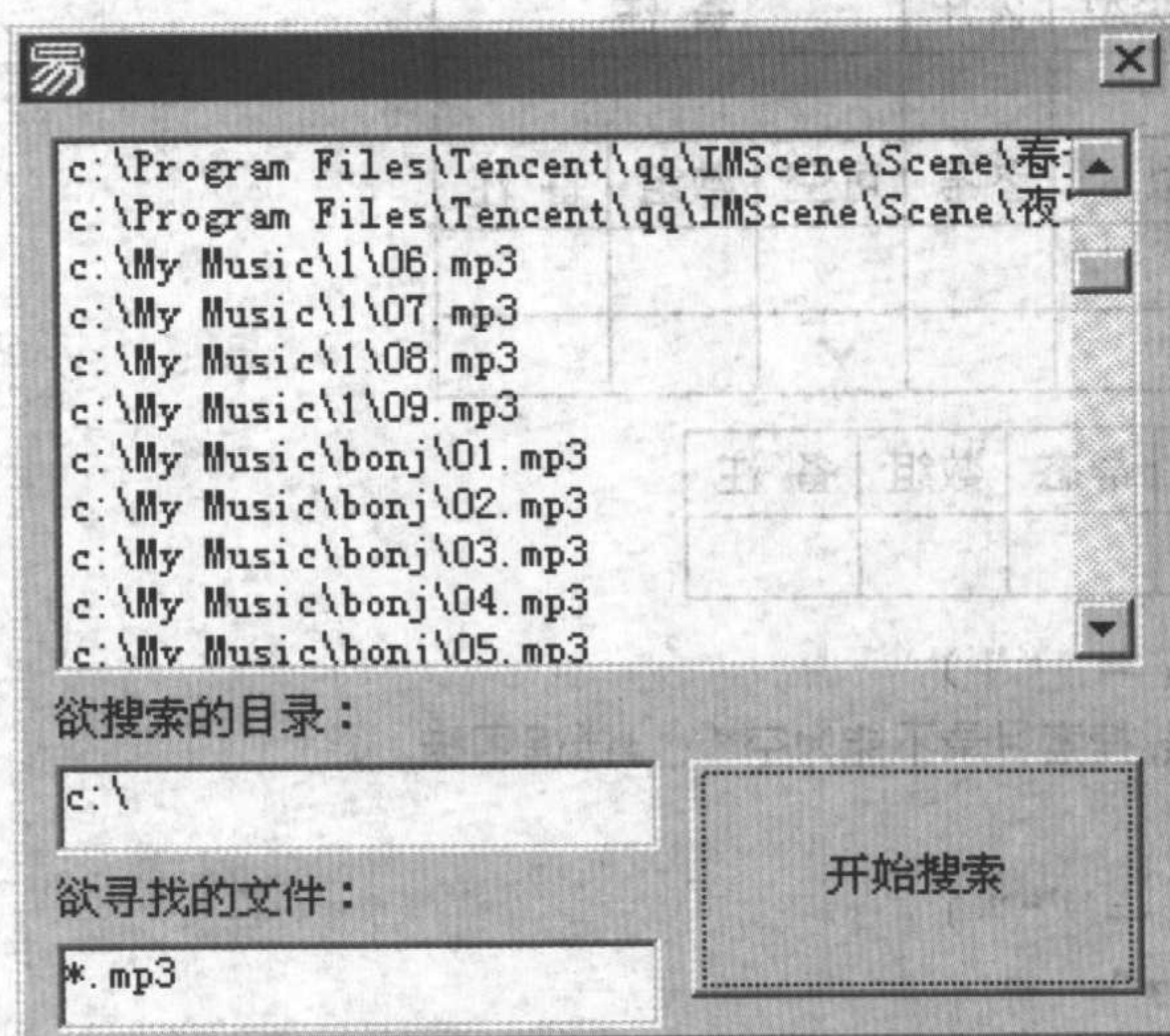


图 3-10 查找到 c 盘所有 mp3 文件

本例程参见随书光盘本章目录中“寻找所有文件.e”



淋病叔味单菜 书盛口窗 章六策

利于程序的编写工作。

系统的事件子程序中的参数最好不要更改。若更改就会被变成用户自定义子程序，从而不能响应事件了。

本章介绍了子程序参数的三个属性“参考”、“可空”及“数组”，概念理解了还需要在实践中多用才能体会到这三个属性的作用。

大家可试着进行以下练习：

1. 熟练掌握子程序的定义方法和参数的用法。
2. 编写一个进行加减乘除运算的子程序,并通过调用该子程序制作一个简单的计算器。
3. 试着新建自定义程序集并在其中编写子程序,然后在其他程序集中调用该子程序。
4. 将第四章的“创建多级目录”例程,改成创建多级目录子程序。返回一个逻辑值表示是否创建成功。

书影口部 12

### 封固本基的窗口 1.1.3

示





## 第六章 窗口组件、菜单和对话框

窗口组件是程序界面可视化操作的基本单元，也是程序中主要的输入/输出部件。所有的组件都是一个独立的对象，拥有自己的属性、事件和方法。易语言可视化的界面设计与良好的对象封装保证了易程序开发的便捷与高效。

菜单作为 Windows 程序特别是早期 Windows 程序的最突出的特点，早已成为用户友好界面必不可少的组成部分。它不仅使用户操作变得异常简单和方便，而且也使程序的界面更加美观。过去，要为一个程序开发一组菜单并不是一件容易的事情。而现在，通过易语言强大的集成开发环境，制作一组美观实用的菜单再也不是一件枯燥繁重的工作。

对话框是封装好的标准样式窗体，实现一个或多个功能。本章将介绍易程序中最常用的“信息框”和“输入框”对话框。

### 6.1 窗口组件

易语言在默认设置情况下，每新建一个易程序窗口程序，都会自动对应生成一个“启动窗口”组件。它有自己的属性、事件和方法，并且可以作为其他控件的载体和容器。在易语言中不能从组件箱中直接新增窗体。新建窗口可使用菜单或在程序面板中使用鼠标右键添加。

#### 6.1.1 窗口的基本属性

为易程序新建一个窗口后，可以在属性面板中查看到该窗口的所有属性。如图 6-1 所示。

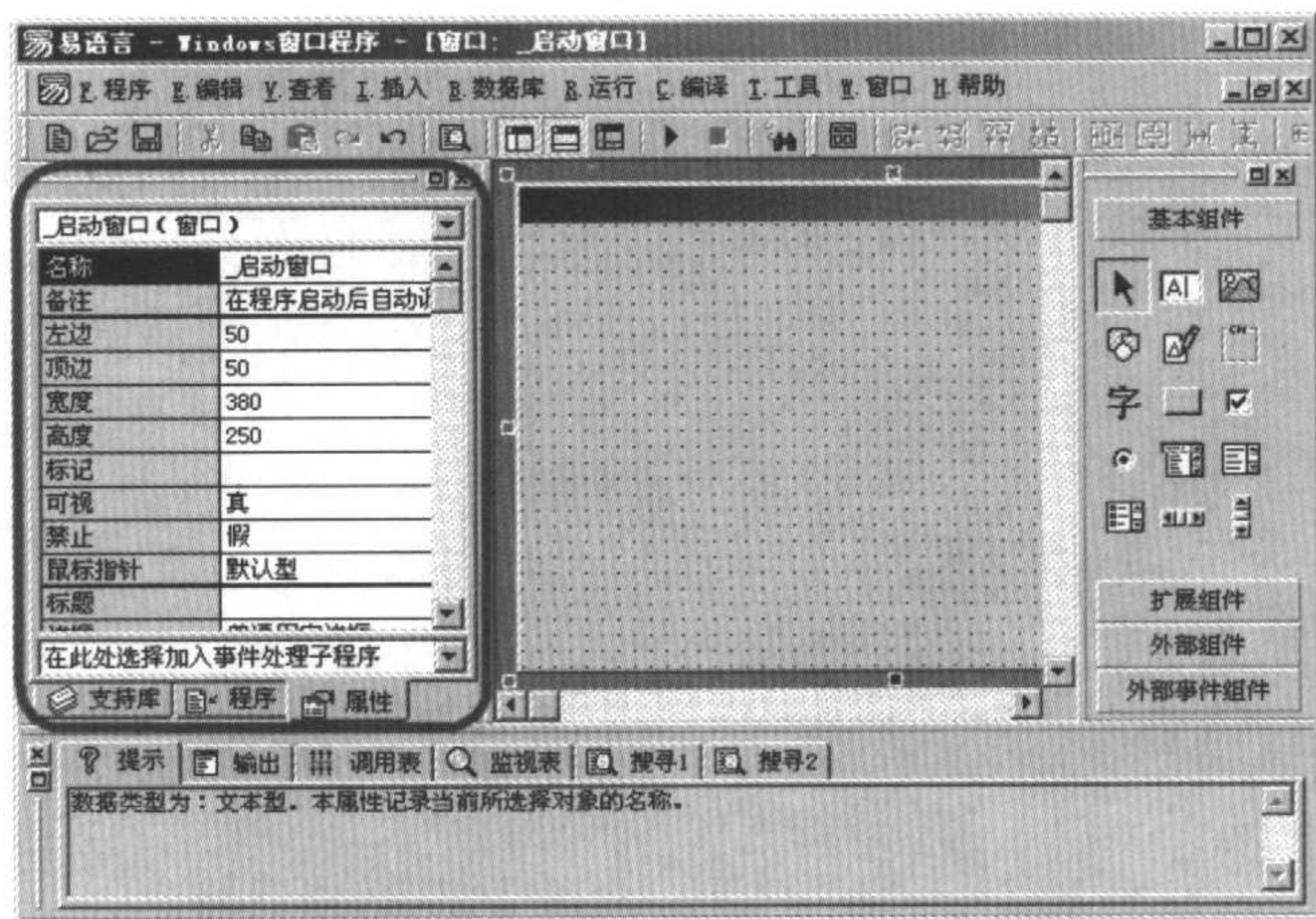


图 6-1 窗口属性



### 1. 手工设置窗口属性

手工设置窗口属性时，只要激活待设置窗口，点击窗口属性面板，修改待调整属性栏的属性值即可。

窗口的基本属性有：“名称”、“标题”、“禁止”、“可视”、“边框”、“左边”、“顶边”、“宽度”、“高度”、“标记”、“备注”等。

例如改变窗口的尺寸，可以在属性面板中“宽度”与“高度”属性栏里输入数值 380，然后回车，此时窗口外形就自动调整为宽与高均为 380 的一个正方形。也可以选中窗体后，直接用鼠标拖动窗体边沿的控制点，将窗体尺寸调整为需要的样式。调整后的窗体宽度和高度会自动反应在属性栏中。

“标题”属性可以设置窗口的标题文字，例如将窗口标题设为“全中文，全可视易语言”。如图 6-2 所示。

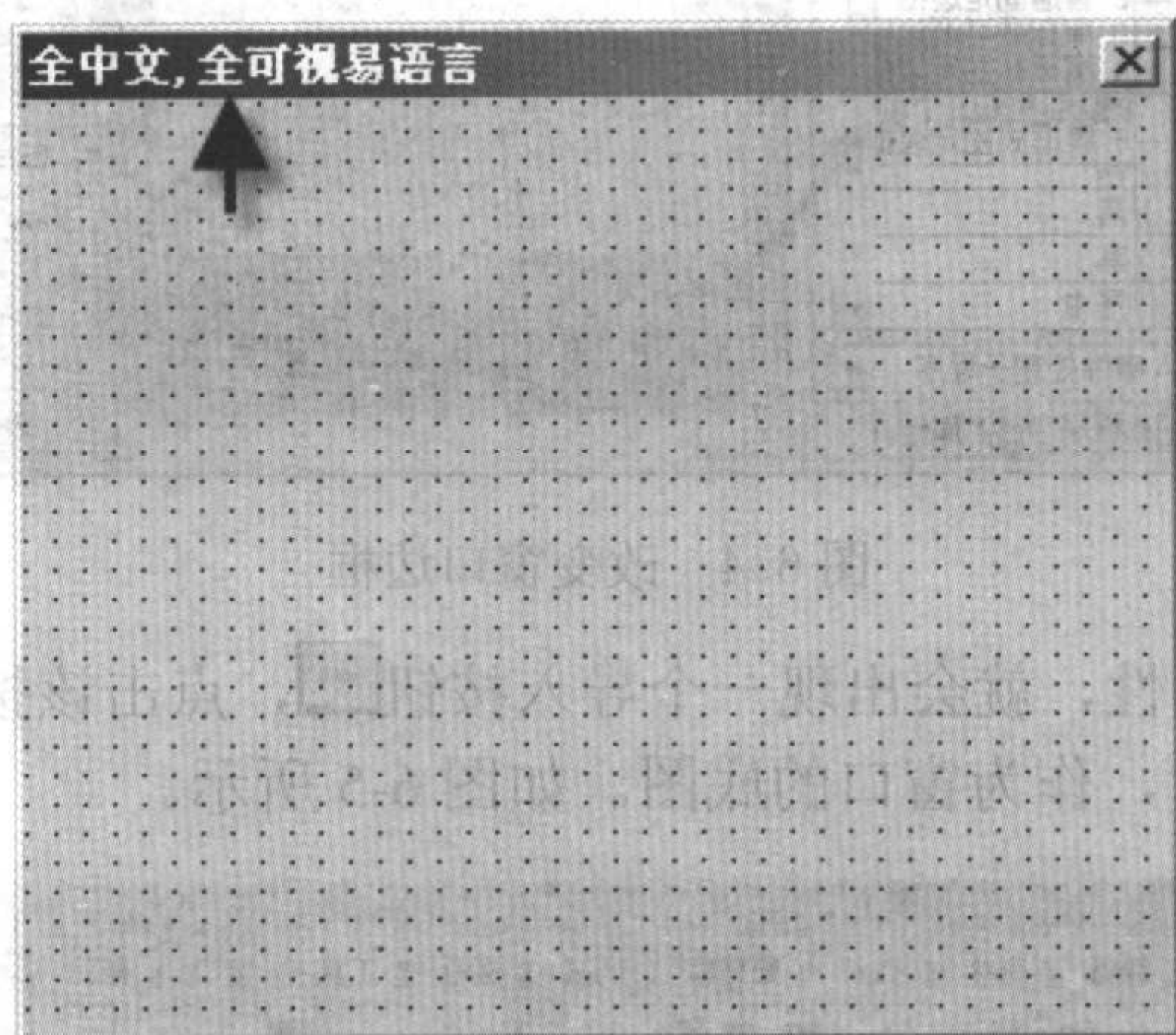


图 6-2 窗口标题属性

增加的新窗口，没有最大化和最小化控制按钮，要添加这些控制按钮，只需将“控制按钮”中的“最大化按钮”、“最小化按钮”属性设为“真”即可。如图 6-3 所示。

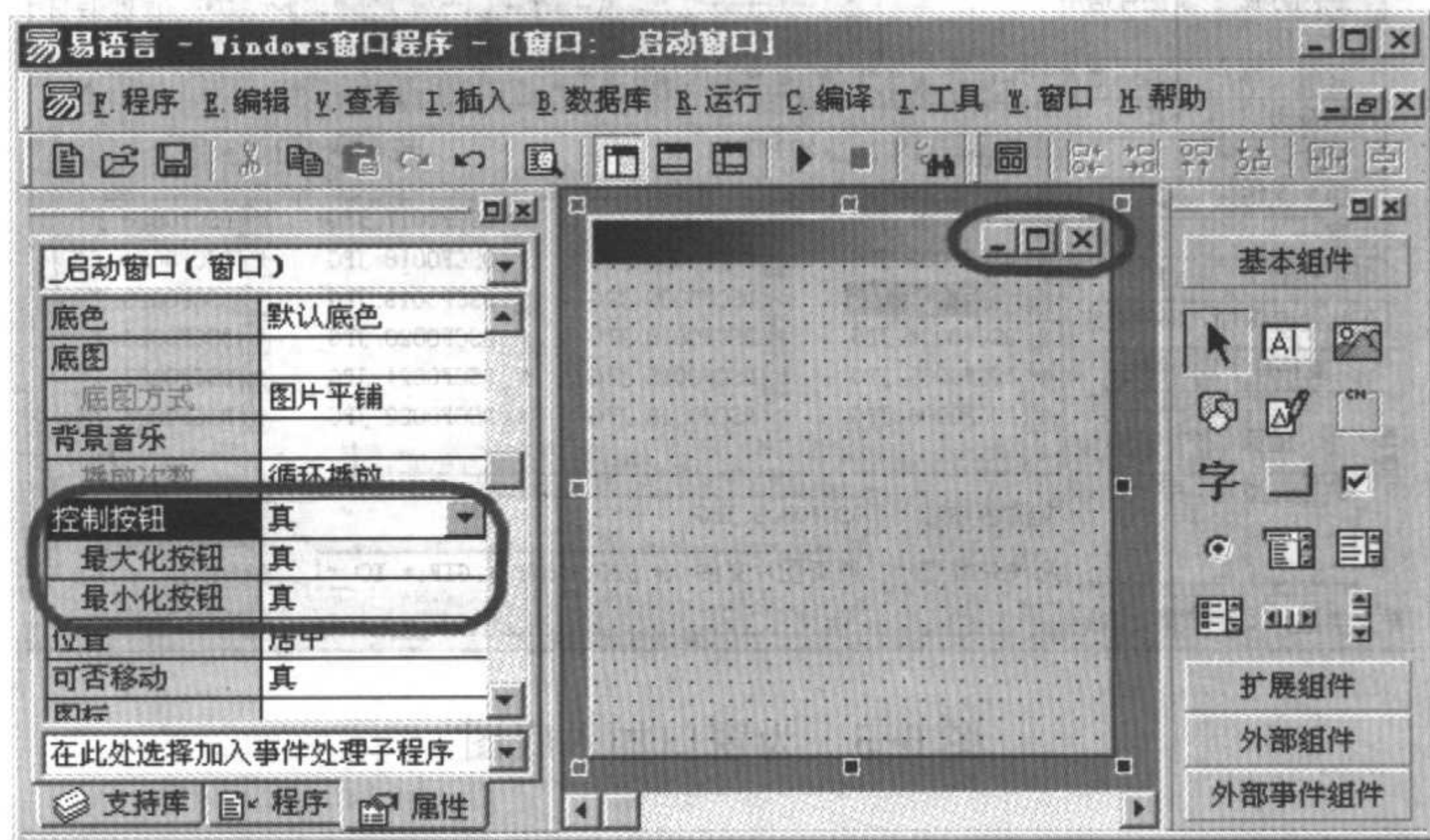


图 6-3 最大化、最小化按钮

除了文字型属性（如“标题”属性）、数值型属性（如“窗口尺寸”属性）、逻辑型属



性（如“可视”属性）还有列表型属性和字节集属性。

列表型属性有“边框”属性、“底图方式”、“播放次数”属性等。例如当大家点击边框属性栏中的下拉按钮时，下拉菜单中有 6 个选项，每一个选项都代表一种不同的边框风格。当窗口边框为“……可调边框”风格时，程序运行后允许自由拖动窗口边缘来改变窗口的大小。如图 6-4 所示。

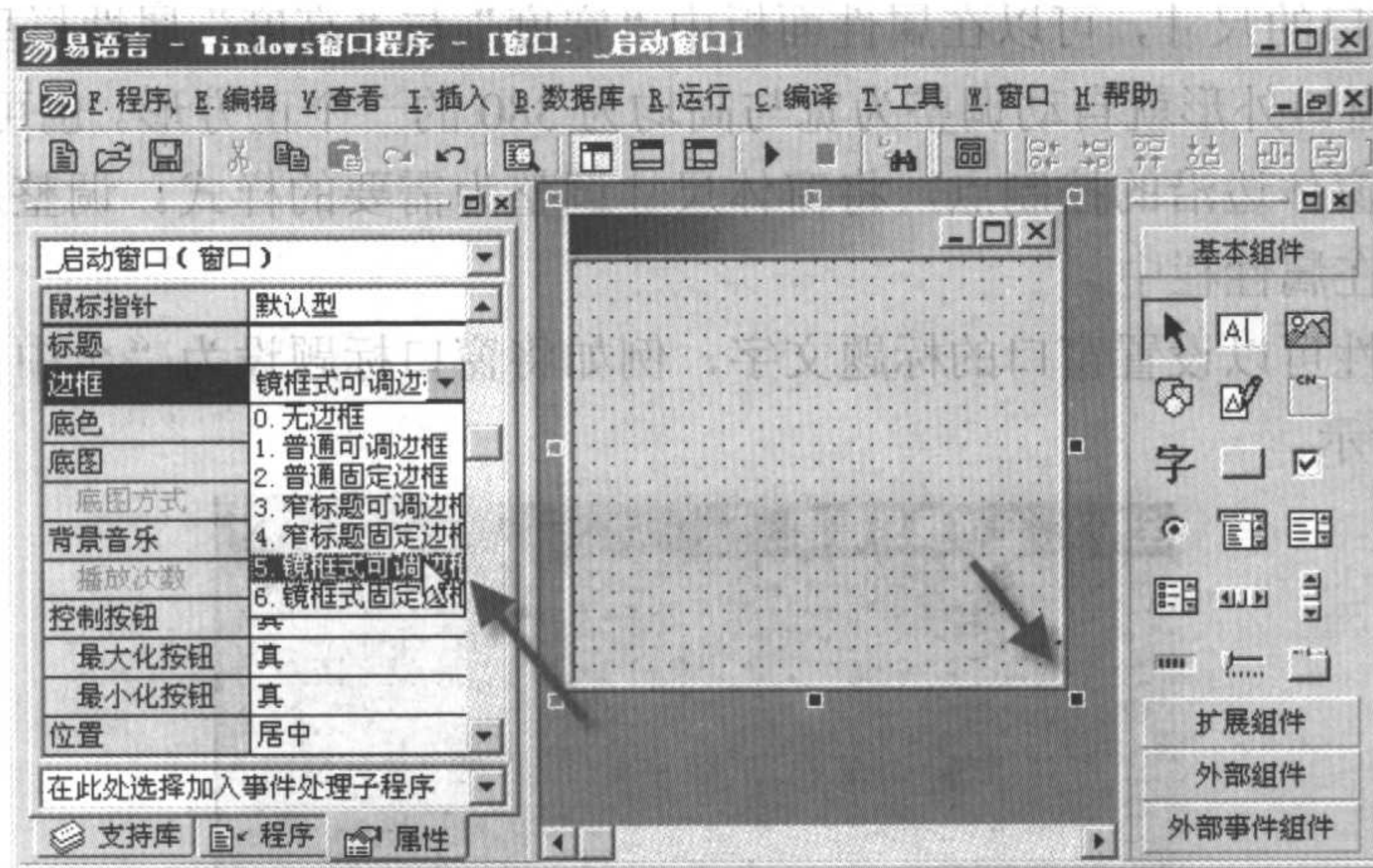


图 6-4 改变窗口边框


选中窗口“底图”属性，就会出现一个导入按钮，点击该按钮就会弹出一个通用对话框，提示选择一张图片，作为窗口的底图。如图 6-5 所示。



图 6-5 为窗口加入底图

“底图”、“背景音乐”、“图标”都是字节集属性，读者可以自行修改其属性内容，实现为窗口加入背景音乐，设置窗口小图标等功能。

字节集属性项中如果存在数据，该属性栏上即会显示“有数据”字样。如欲清除属性



项中的原有数据，可以在该属性项上点击鼠标右键，在弹出菜单中选择“删除内容”，或直接按键盘 Del 键清除。

点击“底色”属性后会弹出调色板对话框，从中可以选择任意一款颜色作为窗口的底色。

以上操作都是在窗口设计时手工设置，在程序运行过程中也可以通过程序代码动态调整各项属性值。

## 2. 在程序运行状态动态调整窗口属性

每个属性都有其相应的数据类型，赋值时需要类型一致。给属性赋值和给变量赋值的方法是相同的。

窗口的边框有以下七种可选样式：“0.无边框”、“1.普通可调边框”、“2.普通固定边框”、“3.窄标题可调边框”、“4.窄标题固定边框”、“5.镜框式可调边框”、“6.镜框式固定边框”等。默认值为“2.普通固定边框”。给边框属性赋值时，只要将欲改变的边框前的数字赋值给该属性即可，例如，将“\_启动窗口”的边框设置为“1.普通可调边框”，代码如下：

```
_启动窗口. 边框 = 1
```

## 3. 窗口其他常用属性

“总在最前”属性，该属性为逻辑型属性，如果设置为“真”，窗口永远显示在屏幕的最前面。默认值为“假”。

“可否移动”属性，如果其值为“真”，可以通过拖动窗口的标题栏来调整窗口的位置。默认值为“真”。

“随意移动”属性，如果其值为“真”，可以通过拖动窗口中任意位置来调整窗口的位置；默认值为“假”。**注意：**如果本属性设置为“真”，则在本窗口中所有不能接收输入焦点的子组件上，均不能再接收到“鼠标左键被按下”和“鼠标左键被放开”事件。

“图标”属性，用于指定显示于窗口标题栏最左端的图标。

“外形”属性，窗口默认的外形是矩形，在外形属性中还提供了 29 种外形可以选择，可以给窗口设置与众不同的外形。

窗口组件的其他属性，请读者自行查看帮助。

## 6.1.2 窗口的基本事件

### 1. 初步了解窗口事件

前面接触到“\_按钮 1\_被单击”事件子程序，该子程序是按钮 1 被单击事件产生的，只要在组件事件子程序中填入相关代码，程序运行时此组件事件被触发后，就会自动运行事件子程序中的代码。

任一种窗口的事件可以在属性面板最下方的事件下拉列表中找到。如图 6-6 所示。



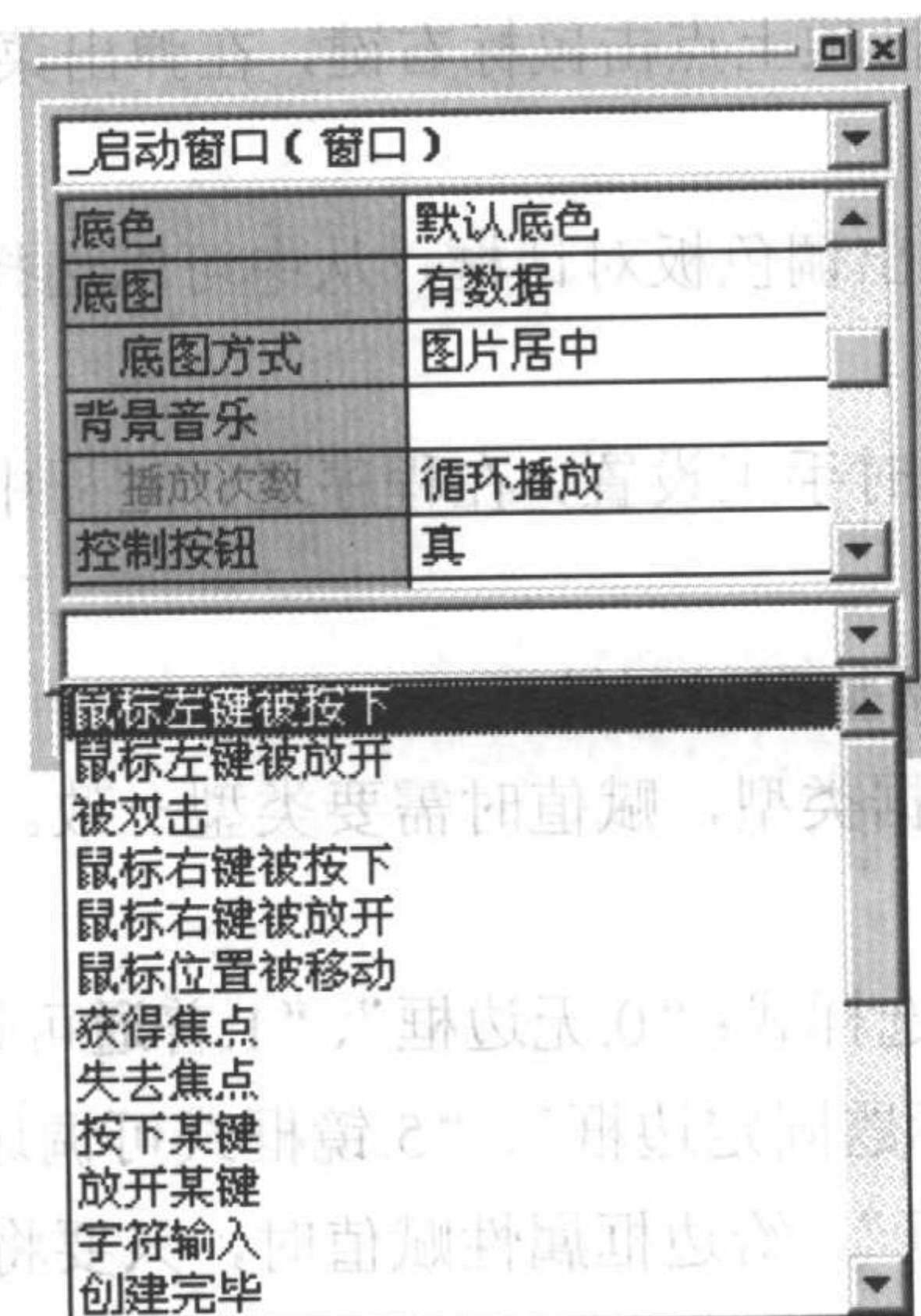


图 6-6 窗口组件的事件

在列表中选择一个事件，就会自动切换到代码编辑界面，并自动生成该组件的事件子程序。

窗口的事件有：“鼠标左键被按下、鼠标左键被放开、被双击、鼠标右键被按下、鼠标右键被放开、鼠标位置被移动、获得焦点、按下某键、放开某键、字符输入、创建完毕、位置被改变、尺寸被改变、将被销毁、可否被关闭、托盘事件、被激活、首次激活、被取消激活、空闲、被显示、被隐藏”等等。

下面编写一个简单的程序，来了解窗口的事件及触发。

首先新建一个易程序，然后在“\_启动窗口”的事件下拉列表中选择“鼠标左键被按下”。然后在“\_启动窗口\_鼠标左键被按下”事件子程序中输入代码：

子程序名	返回值类型	公开	备注		
_启动窗口_鼠标左键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

信息框 (“鼠标左键被按下”, 0, )

运行程序，可以看到每次在窗体上点击鼠标左键，都会弹出信息框提示：“鼠标左键被按下”。

## 2. 窗口的常用事件

### (1) “创建完毕”事件

事件的产生时机：当窗口及其中的所有组件均被创建完成，即将显示之前，将触发此事件。**注意：**是在窗口创建但未显示之前触发本事件。



“创建完毕”事件是窗口最重要、也是最常用的事件。通常在本事件的“事件处理子程序”中做一些关于本窗口的初始化工作。“\_启动窗口”的“创建完毕”事件尤为重要，通常在此处完成整个程序的初始化工作。

(2) “位置被改变”事件


事件的产生时机：当窗口的位置被改变时产生此事件。

(3) “尺寸被改变”事件

事件的产生时机：当窗口的尺寸被改变时产生此事件。

(4) “将被销毁”事件

事件的产生时机：当窗口被销毁之前产生此事件。

用户单击窗口右上角的按钮，或执行到命令代码“销毁()”时，就会触发此事件。如果本事件子程序中存在程序代码，则先执行它们，再销毁窗口。

编程者通常在本事件子程序中做一些收尾工作，如保存当前窗口的位置与尺寸，下次运行程序时就可以自动以本次最终显示方式显示窗口了。

(5) “可否被关闭”事件

事件的产生时机：在窗口关闭之前产生此事件，稍早于“将被销毁”事件。

本事件有一个逻辑型返回值。如果在本事件的“事件子程序”中返回“假”，表示不允许关闭窗口；如果返回“真”或不返回值，表示允许关闭窗口。

例如：检查当前被编辑文档有没有存盘，如果没存盘，显示对话框提示文档尚未存盘。这样，用户在运行程序时就可以选择放弃关闭窗口的操作。

(6) “托盘事件”事件

事件的产生时机：当用户用鼠标单击或双击任务栏托盘中的图标（用“置托盘图标()”命令设置）时产生本事件。

本事件有一个整数型参数“操作类型”，其值可能是如下之一：“1.#左键单击”；“2.#左键双击”；“3.#右键单击”。程序可根据此参数判断出用户的击键动作，实现相应的处理，比如是弹出菜单还是显示窗口等。

(7) “被激活”、“首次激活”、“被取消激活”事件

事件的产生时机：当窗口被激活、首次激活或被取消激活时产生本事件。

窗口的标题栏底色由灰色变为蓝色时，表示窗口“被激活”；由蓝色变为灰色时，表示“被取消激活”；“首次激活”即窗口被显示后第一次被激活。“首次激活”事件在窗口被销毁之前只会产生一次，而“被激活”、“被取消激活”事件可能产生很多次。

(8) “空闲”事件

事件的产生时机：当系统（CPU、操作系统或程序）空闲时产生本事件。

本事件有一个整数型参数“已空闲时间”，用于告知编程者事件产生时，系统已经进入空闲状态的时间，单位为毫秒。如果需要在本事件被触发时实现额外功能，应该在“已空闲时间”的值较大以后再进行。

本事件有一个逻辑型的返回值。如果事件处理子程序返回“假”或者不返回值，在此次空闲期间系统将不再产生空闲事件（可以降低 CPU 占用率）。如果返回“真”，系统将产生空闲事件。



- (9) “被显示”、“被隐藏”事件
- 事件的产生时机：当窗口被显示、被隐藏（比如令“可视”属性为假）时触发本事件。
- “被显示”事件在窗口被销毁之前至少产生一次；“被隐藏”事件可能一次也不产生。

### 6.1.3 增加新窗口和弹出窗口

1. 增加新窗口
- 增加新窗口的方法可以通过点击易语言主菜单：“插入”→“新窗口”，插入的窗口将被自动命名为“窗口 1”、“窗口 2”……编程者可改变属性面板中的“名称”属性，为它们重新命名。建议给窗口起一个有意义的名称，便于管理，如“欢迎窗口”、“登录窗口”等等。
- 也可以选中程序面板，在窗口栏中点击鼠标右键，通过右键菜单中的“插入新窗口”菜单项来插入新窗口。右键菜单中还有“删除窗口”菜单项，可以删除当前被选中的窗口。如图 6-7 所示。

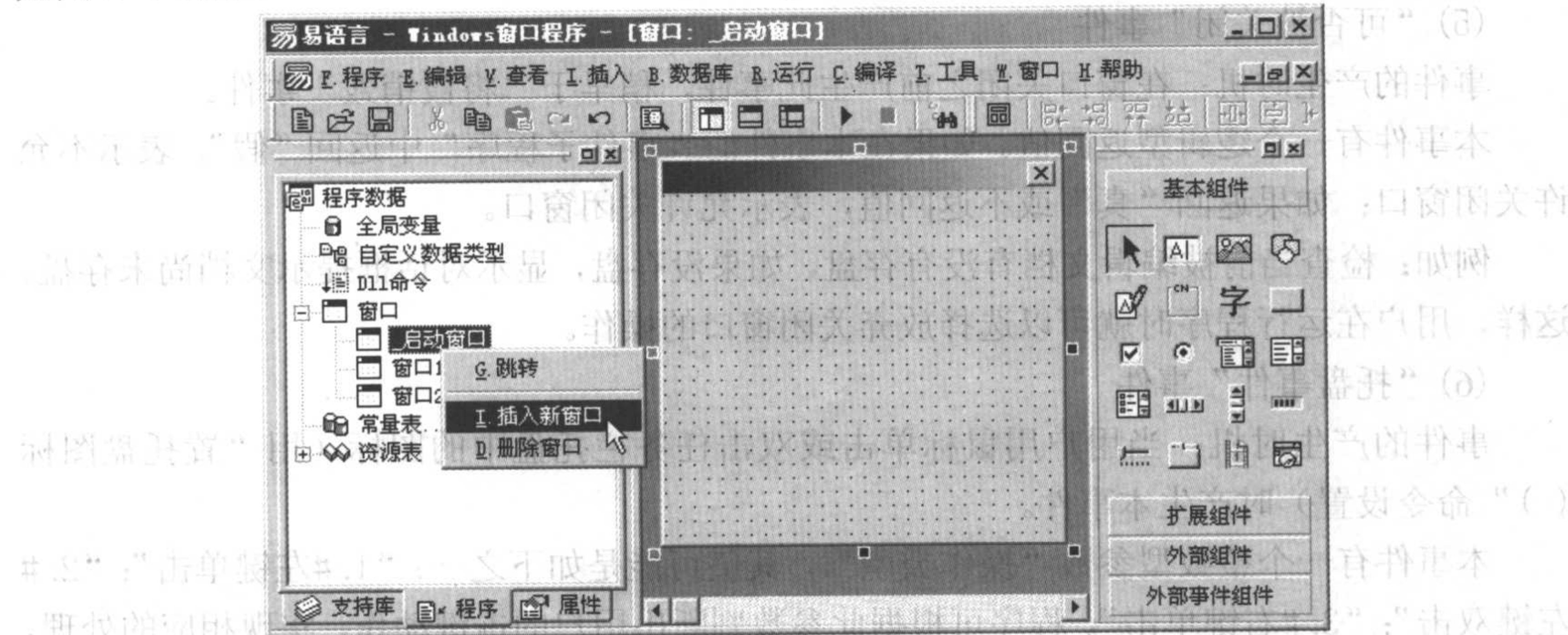


图 6-7 插入新窗口

2. “载入 ( )”命令载入新窗口
- 当新增加了一个窗口后，若想在程序运行过程中显示出来，必须要使用“载入 ( )”命令。如：新建一个易程序，生成一个“\_启动窗口”，上面放了一个按钮组件，新建了一个窗口 1，按钮被单击事件程序代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

载入 (窗口1, , 真)

- 上述程序运行后，可以通过点击启动窗口中的按钮弹出窗口 1。
- “载入 ( )”命令有 3 个参数，第一个参数指定欲载入的窗口。第二个参数指定欲载入窗口的父窗口，此父窗口必须已经被载入。可以被省略。第三个参数为逻辑型参数，确定新窗口被载入后，是否允许同时对其他窗口进行操作。如果本参数值为真：(1) 即使被载



入窗口的“可视”属性为假，在载入后也立即显示；(2) 本命令一直等到该窗口被销毁后才返回。

下面制作一个简单的登录窗口，来加深对多窗口的认识。

首先，新建一个易程序，将“\_启动窗口”的可视属性设置为“假”，然后插入一个新窗口，将插入的“窗口1”的名称属性改为“登录窗口”。

然后，在“登录窗口”中添加编辑框组件、标签组件和按钮组件各两个。标签组件的标题改为“用户名”和“密码”，按钮组件的标题改为“登录”和“取消”。如图 6-8 所示。

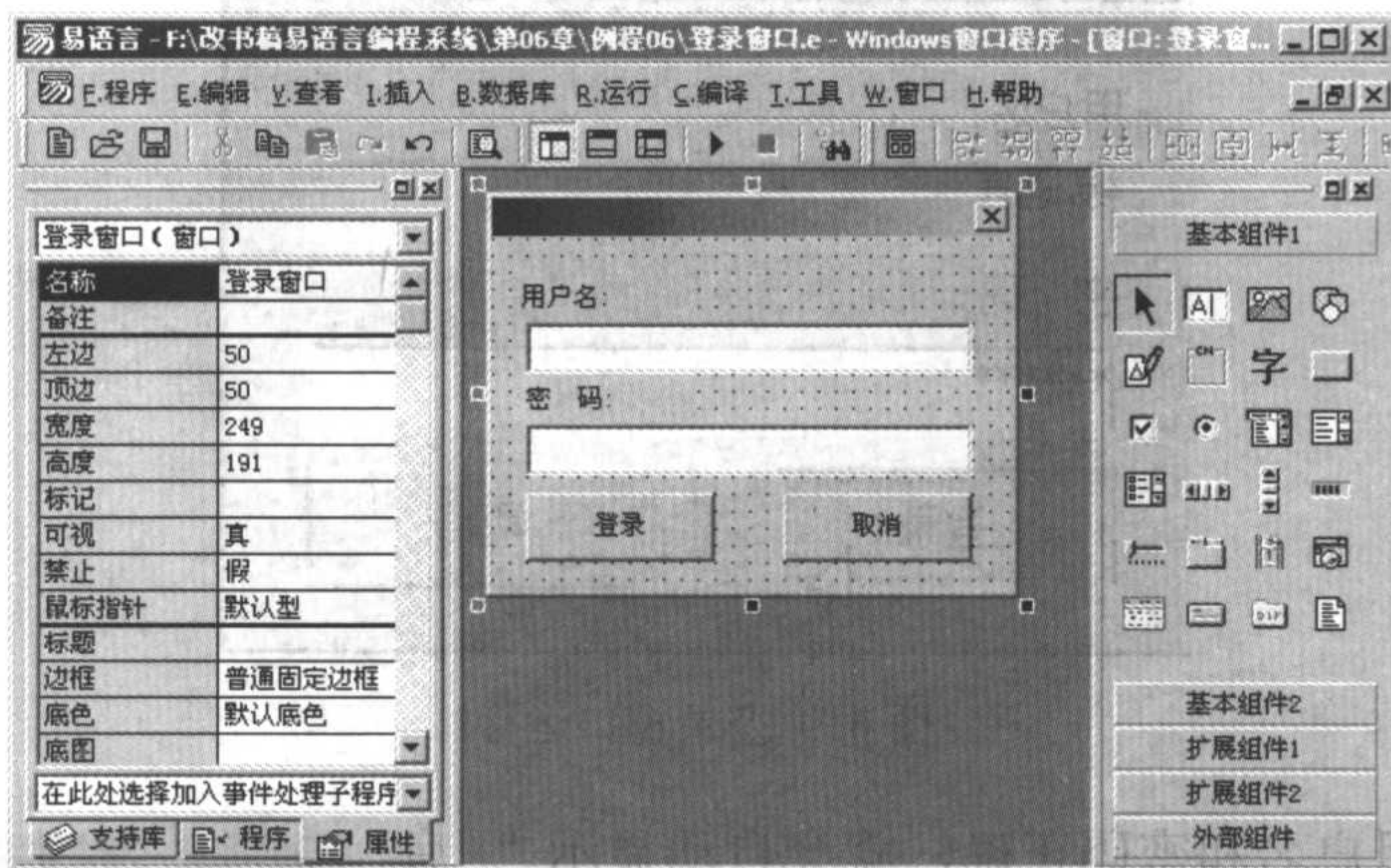


图 6-8 登录窗口界面

将用来输入密码的编辑框的“输入方式”属性改为“密码输入”，然后双击“登录”按钮，在“\_按钮1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```

--- 判断 (编辑框1.内容 = “易语言”)
    --- 如果 (编辑框2.内容 = “123456”)
        信息框 (“登录成功”, 0, )
        登录窗口.销毁 ()
    --- 信息框 (“密码错误”, 0, )
    --- 信息框 (“用户名错误”, 0, )
    
```

在点击“登录”按钮后，就对输入的用户名和密码进行判断，这里假设用户名为“易语言”，密码为“123456”。如果用户名和密码都正确，就弹出信息框提示，并销毁“登录窗口”；如果某一项出现错误，则对话框显示相应的错误提示。

回到“登录窗口”，双击“取消”按钮，在“\_按钮2\_被单击”子程序中输入代码：

结束 ()

在程序面板点击“\_启动窗口”，切换到“\_启动窗口”中，双击“\_启动窗口”，在产生





的“\_启动窗口\_创建完毕”子程序中输入代码：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

载入 (登录窗口, \_启动窗口, 真)

最后，试运行程序，程序运行后，会直接弹出“登录窗口”，在登录窗口中输入用户名和密码，点击“登录”，如图 6-9 所示。

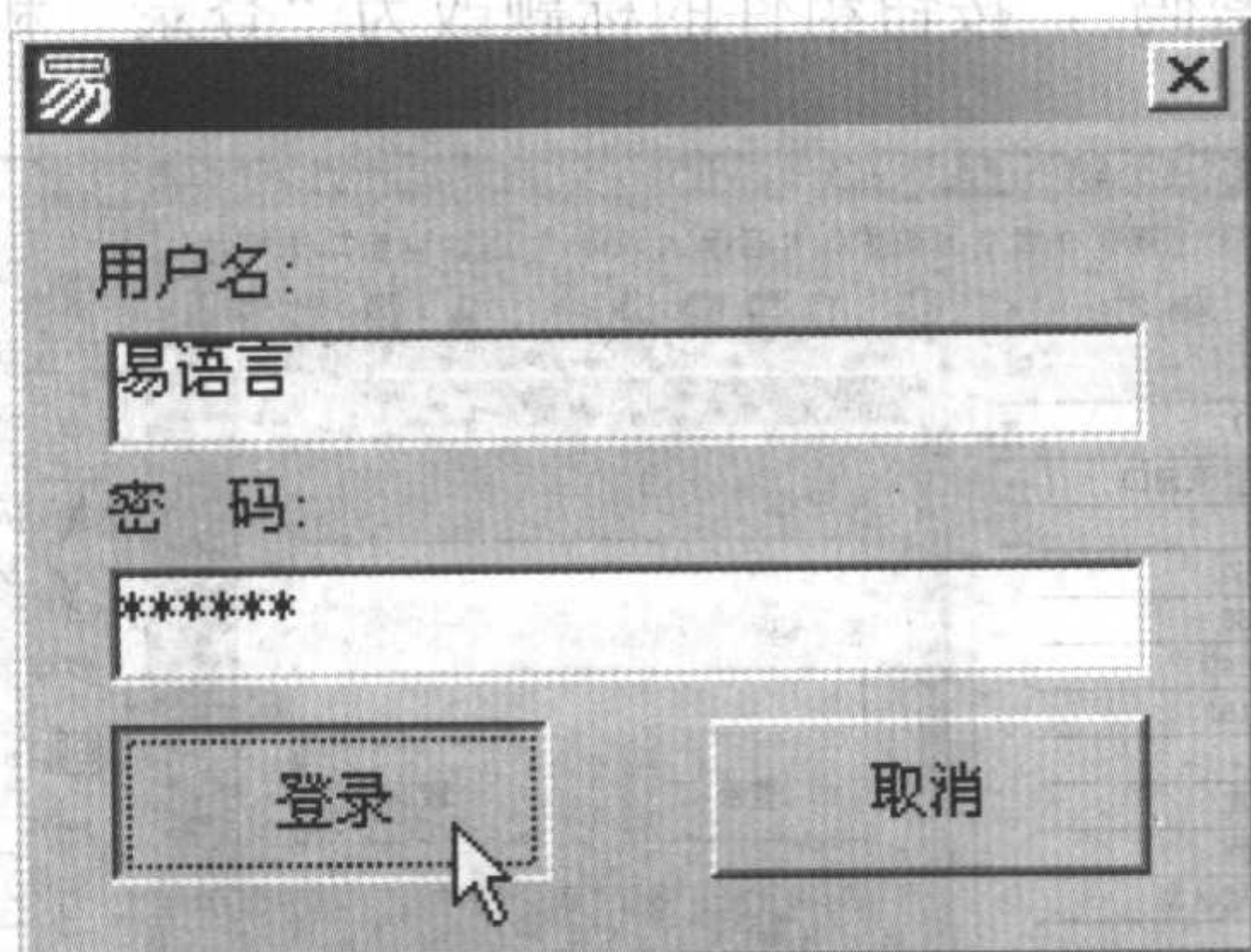


图 6-9 登录窗口运行

如果输入的用户名和密码正确，就会自动显示“\_启动窗口”。

本例程参见随书光盘中的“登录窗口.e”。

## 6.1.4 窗口的重要方法

窗口的重要方法有：“销毁()”、“激活()”、“置托盘图标()”、“弹出托盘菜单()”等。

窗口的所有方法都可以用以下方法查找到：打开左侧的支持库面板，展开“系统核心支持库” → “数据类型” → “窗口”中，其中列出了所有的方法。如图 6-10 所示。

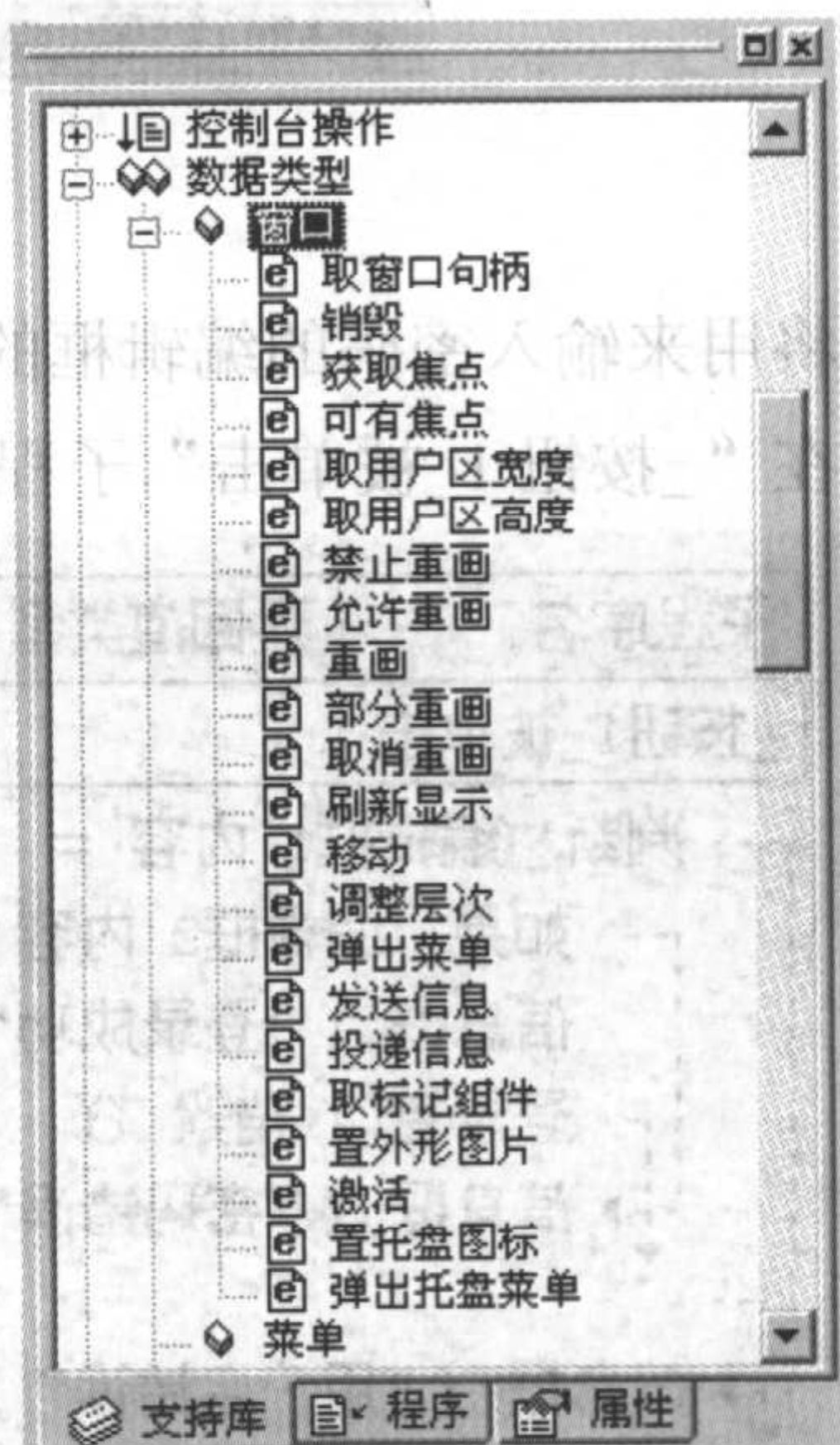


图 6-10 窗口的所有方法

### 1. “移动()”方法

本方法用来改变窗口或窗口组件的位置和尺寸，有 4 个参数，分别为指定窗口的“左边”、“顶边”、“宽度”、“高度”。例如，将“窗口 1”移动到左边 20、顶边 20，尺寸改变为宽度 100、高度 200，可以使用如下代码：

窗口 1. 移动 (20, 20, 100, 200)

也可以通过为“窗口 1”属性赋值的方法达到相同的效果：

窗口 1. 左边 = 20



窗口 1. 顶边 = 20

窗口 1. 宽度 = 100

窗口 1. 高度 = 200

虽然两者效果相同，但一个是调用了组件方法实现，而另一个是为组件属性赋值。

## 2. “销毁 ()”方法

该方法可以将指定窗口关闭，并释放该窗口的所有资源，如：

\_启动窗口.销毁 ()

“\_启动窗口”被销毁后，程序也就自动结束了。而销毁普通窗口（非“\_启动窗口”），不会影响程序继续运行。

窗口的“销毁 ()”方法与“结束 ()”命令不同。“结束 ()”命令会结束整个程序的运行；而“销毁 ()”只是关闭指定的窗口并释放窗口资源（仅当“\_启动窗口”被销毁后，才会结束程序）。

## 3. “激活 ()”方法

激活当前窗口。激活后的窗口位于系统前台，拥有输入焦点，其标题栏变为蓝色。例如以下可将“\_启动窗口”激活：

\_启动窗口.激活 ()

如果在窗口未激活状态下调用本命令，将会激活窗口，并产生“被激活”事件；如果在窗口激活状态下调用本命令，不会产生“被激活”事件。

## 4. “置托盘图标 ()”方法

该方法可以设置本程序在系统托盘中的图标。本方法有 2 个参数。

参数 1：图标数据——指定出现于托盘中的图标，其值可为图标资源、图标字节集数据或图标文件名，如果省略本参数，表示清除已经设置了的托盘图标；

参数 2：提示信息——当把鼠标放到托盘图标上时，出现的提示文本。

下面就来制作一个有托盘图标的窗口。

新建一个易程序，然后在资源表中添加一个新图片资源，名为“托盘图标”，在该资源中加入一个图标文件（.ICO 文件）。

要注意“置托盘图标 ()”的图片不能过大，最好是图标文件或尺寸大小类似的图片，图片过大会导致“置托盘图标 ()”方法失败。

然后，在窗口中添加一个按钮组件，将按钮组件的标题改为“隐藏窗口”，然后双击按钮，在“\_按钮 1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

\_启动窗口.可视 = 假

\_启动窗口.置托盘图标 (#托盘图标, “置托盘图标例程”)





回到“\_启动窗口”，选择“\_启动窗口”事件中的“托盘事件”，在“\_启动窗口\_托盘事件”子程序中输入代码：

子程序名	返回值类型	公开	备注		
__启动窗口_托盘事件					
参数名	类型	参考	可空	数组	备注
操作类型	整数型				

```
如果真 (操作类型 = 2)
    置托盘图标 ({ }, )
    _启动窗口.可视 = 真
```

当托盘事件的“操作类型”为 2，即在托盘图标上双击鼠标左键，就清除托盘图标，并显示“\_启动窗口”。

最后，试运行程序，点击“隐藏窗口”按钮。如图 6-11 所示。

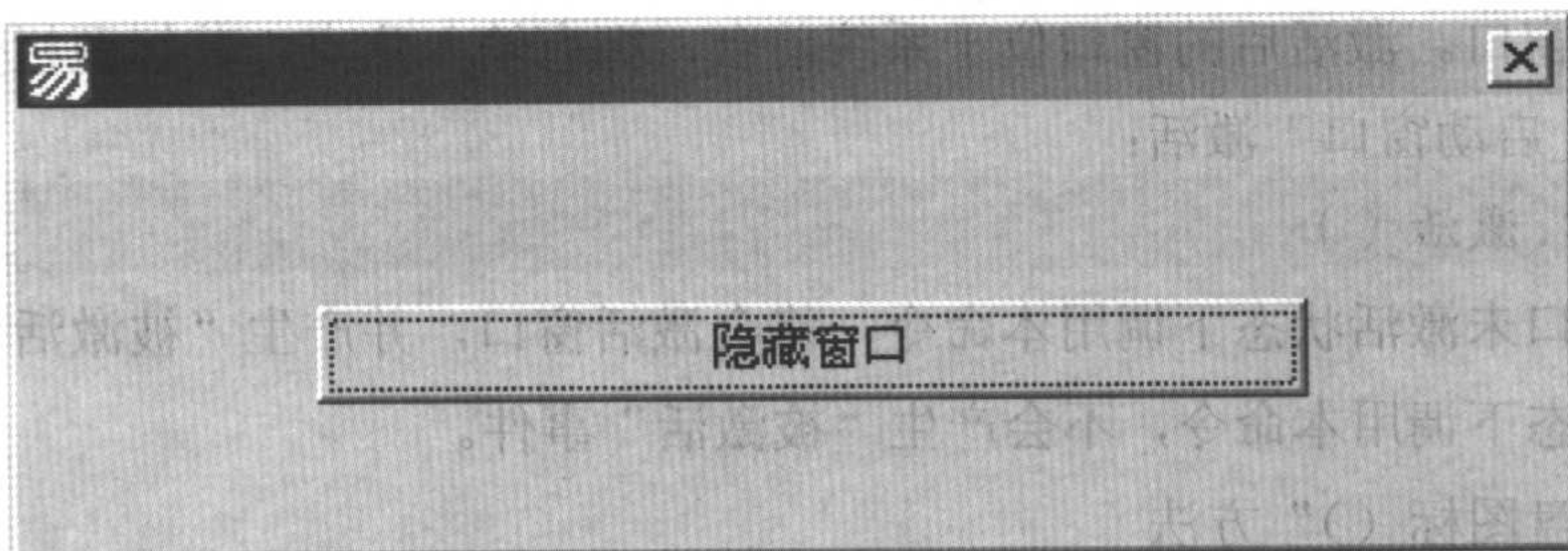


图 6-11 隐藏窗口例程

窗口被隐藏后，可以在系统托盘区看到该窗口的托盘图标。如图 6-12 所示。

置托盘图标例程



图 6-12 托盘图标

双击托盘图标，可以看到窗口恢复显示，且托盘图标也被清除。本例程参见随书光盘中的例程“置托盘图标.e”。

#### 5. “弹出托盘菜单 ()”方法

该方法用来弹出托盘菜单。参数中添入欲弹出菜单名称即可。

在上面“隐藏窗口”例程中，添加一个新菜单。

在“\_启动窗口”中点击鼠标右键，选择“菜单编辑器”，会弹出“菜单编辑器”的对话框，在对话框中插入一个菜单项及 2 个子菜单，主菜单名为“托盘菜单”，2 个子菜单分别为“显示窗口”、“关闭程序”，如图 6-13 所示。



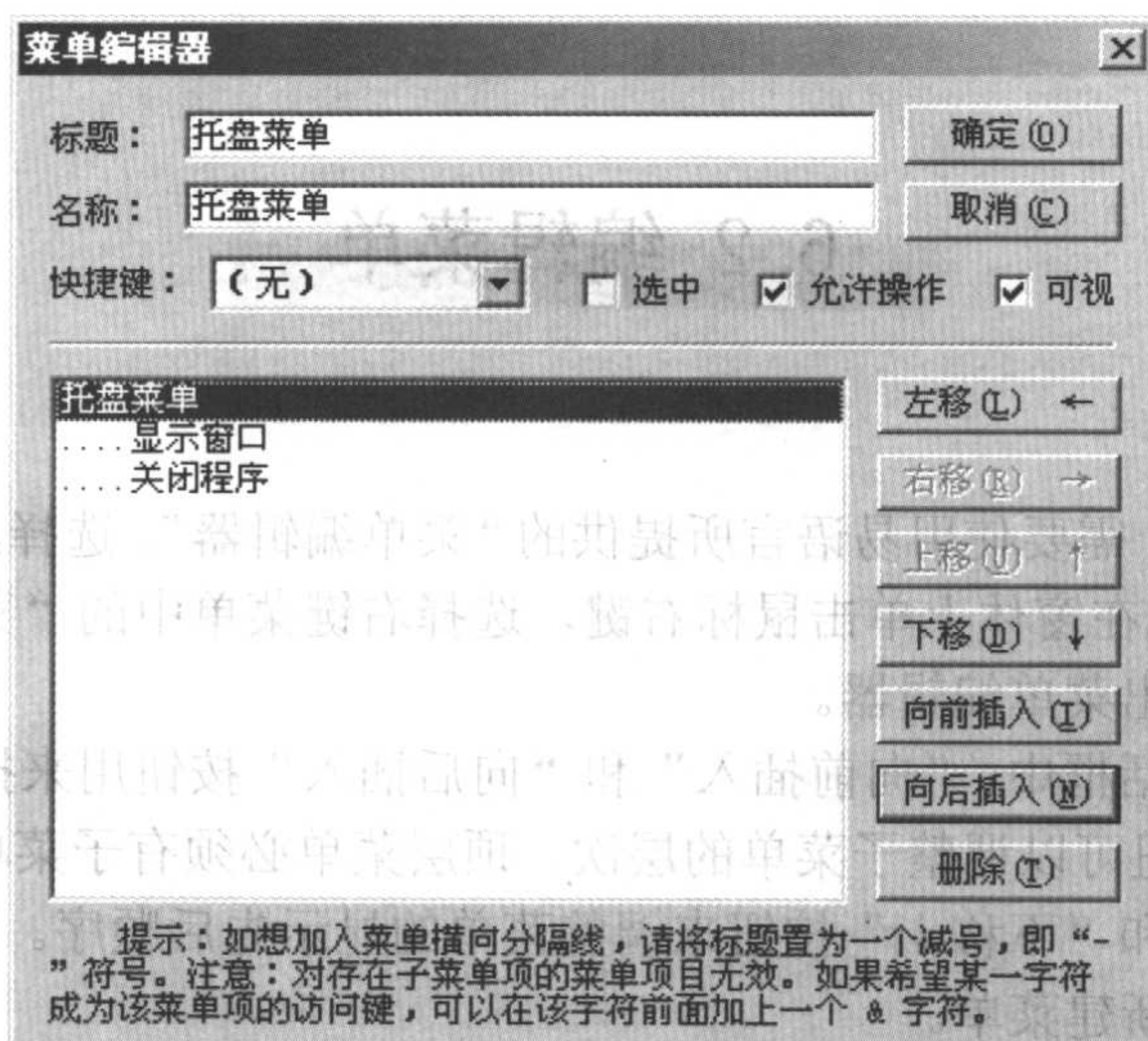


图 6-13 易语言菜单编辑器

添加了一个菜单后，可以在启动窗口中看到该菜单，如图 6-14 所示。

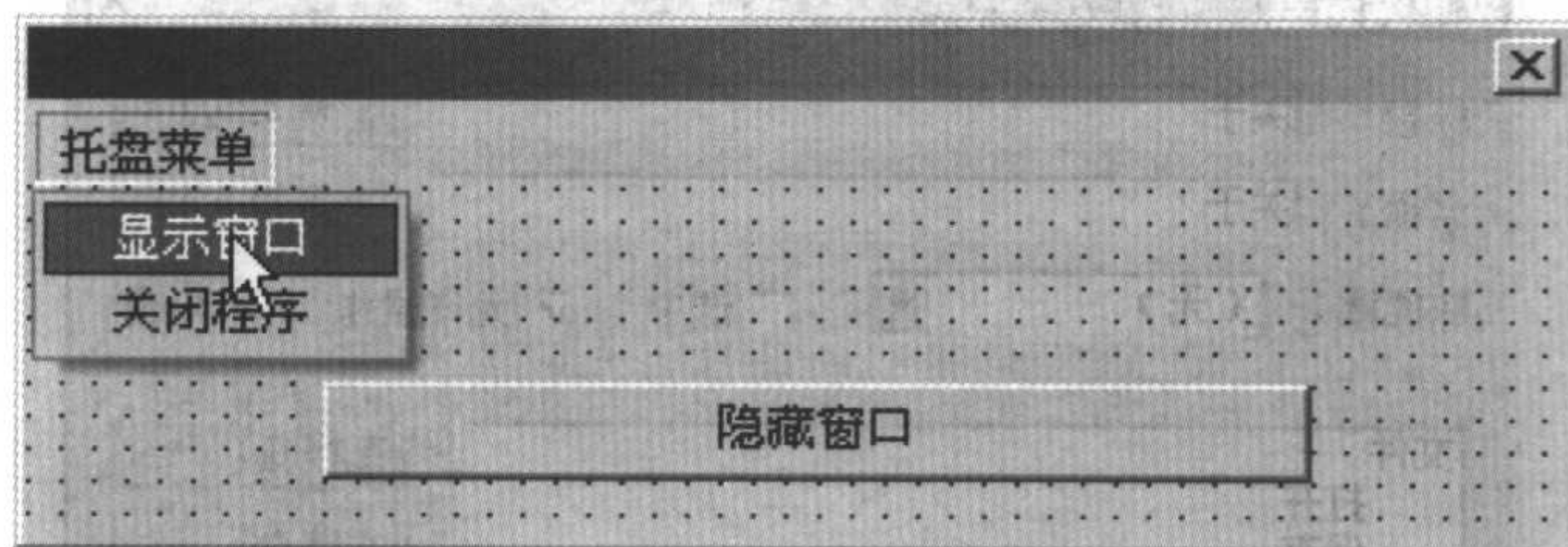


图 6-14 添加菜单后的窗口

然后在“\_\_启动窗口\_托盘事件”子程序中继续输入以下代码：

```

如果真 (操作类型 = 3)
    启动窗口.弹出托盘菜单 (托盘菜单)
    
```

判断当托盘事件的操作类型为“3”时(即在托盘图标上单击鼠标右键)，就用“弹出托盘菜单 ()”的方法弹出启动窗口中的“托盘菜单”。

最后试运行程序，在托盘图标上单击鼠标右键。如图 6-15 所示。

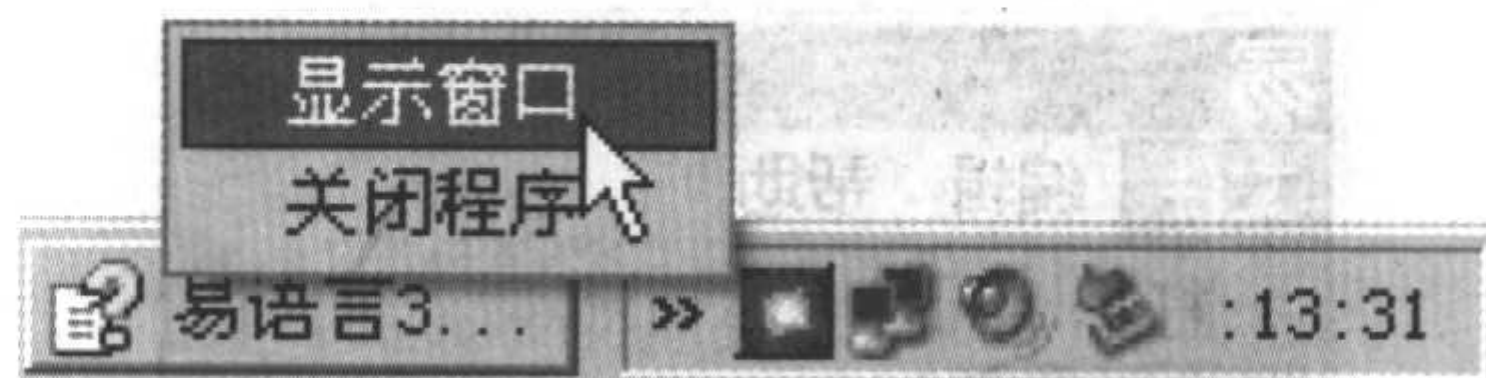


图 6-15 弹出托盘菜单

点击菜单可以看到没有任何效果，是因为还没有给菜单编写任何代码，对菜单的编辑将在下节介绍。





## 6.2 编辑菜单

### 6.2.1 新建菜单

为程序添加菜单，需要使用易语言所提供的“菜单编辑器”。选择易语言主菜单“工具”→“菜单编辑器”；或在窗体上单击鼠标右键，选择右键菜单中的“菜单编辑器”；还可以使用热键[Ctrl+E]调出菜单编辑器。

在菜单编辑器对话框中，“向前插入”和“向后插入”按钮用来插入新的菜单，“左移←”和“右移→”按钮可以调整子菜单的层次。顶层菜单必须有子菜单，否则会出现错误。可以点击“上移↑”和“下移↓”按钮来调整菜单的显示先后顺序。

下面具体讲解如何新建菜单。

新建一个易程序，打开菜单编辑器，在编辑器中新建 3 个顶层菜单“文件”、“编辑”、“帮助”，然后给每个顶层菜单加入子菜单。如图 6-17 所示。

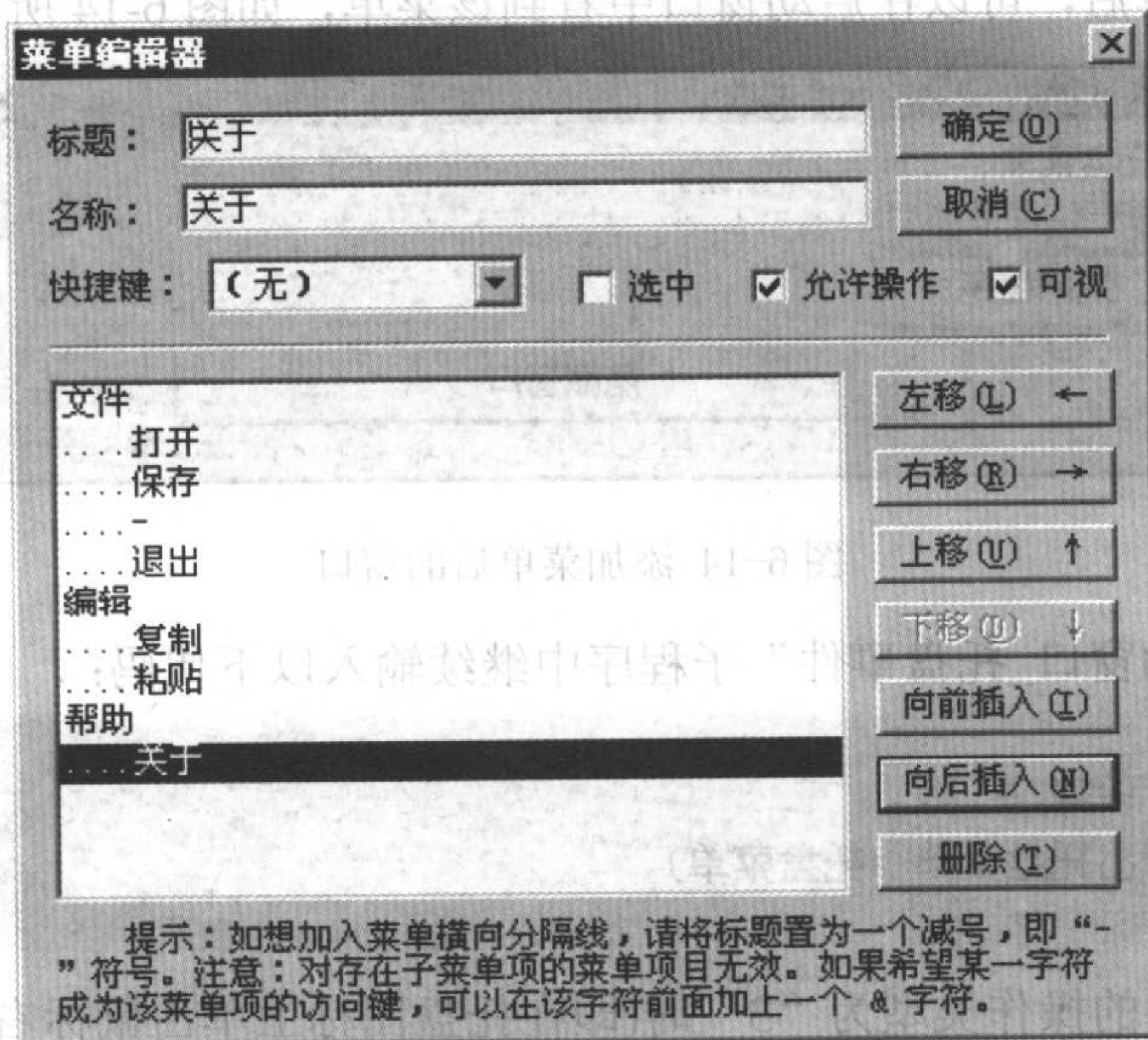


图 6-17 编辑菜单

可以试运行程序，查看新建的菜单，如图 6-18 所示。

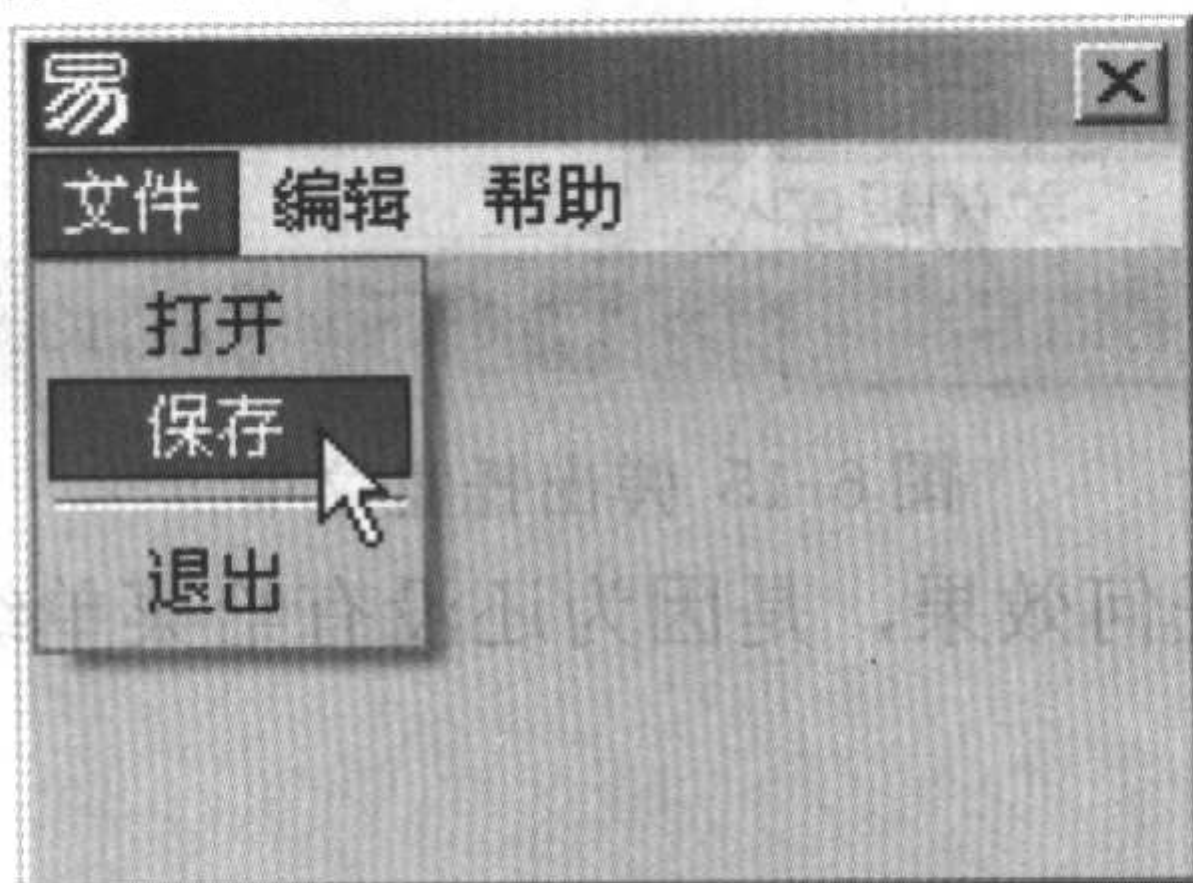


图 6-18 菜单效果



菜单中“保存”和“退出”选项之间的分隔线，是菜单项“-”。

## 6.2.2 菜单的热键及属性

### 1. 菜单的热键

可以为菜单项设置快捷键。即按住 Alt 键后，同时按下另一个键，就可以调用指定的菜单项功能。一般情况下，有两种表示方法：

**菜单名(&F) 或 &F.菜单名**

例如将刚才编辑过的菜单都加上热键。如图 6-19 所示。

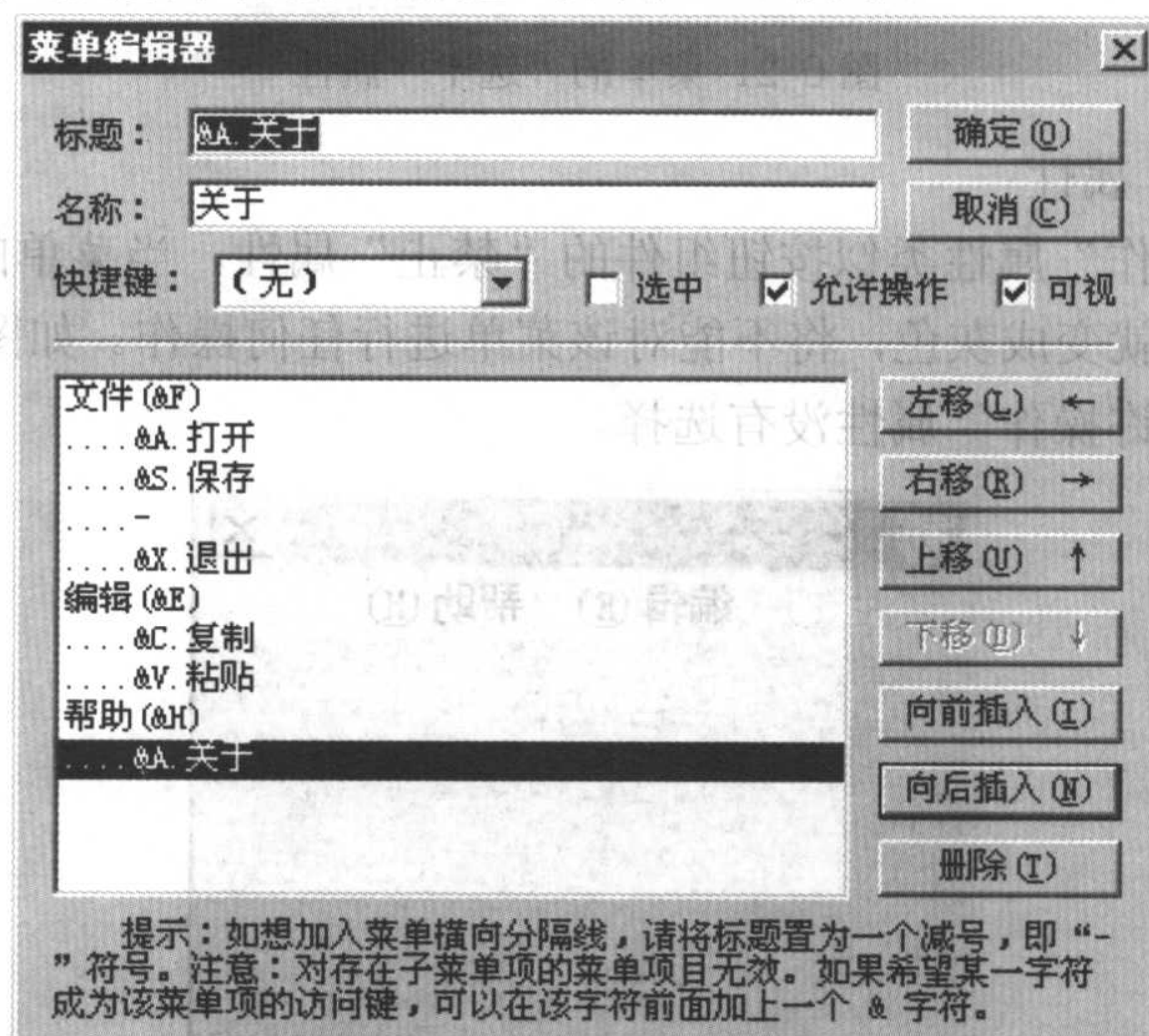


图 6-19 菜单加热键

运行程序后，可以使用自定义的热键来打开菜单，如图 6-20 所示。

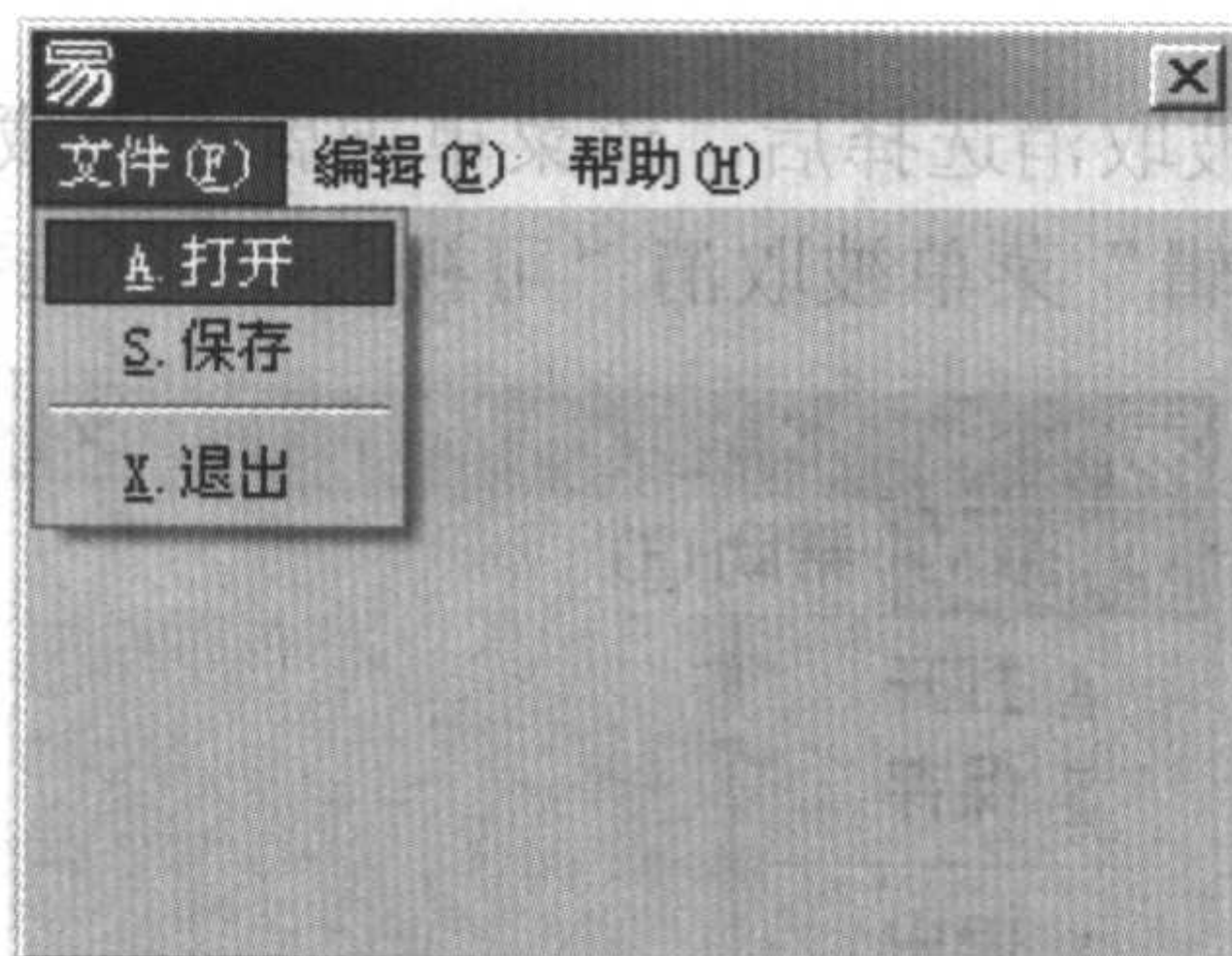


图 6-20 用热键打开菜单

### 2. 菜单的属性

在编辑菜单项时会发现，菜单有“选中”、“允许操作”及“可视”这 3 个属性。

#### (1) “选中”属性

可以控制菜单是否为可选菜单，一个菜单“选中”属性被选择后，该菜单的前面就会出现“√”。如图 6-21 所示，图中的“关于”菜单的“选中”属性被选择。



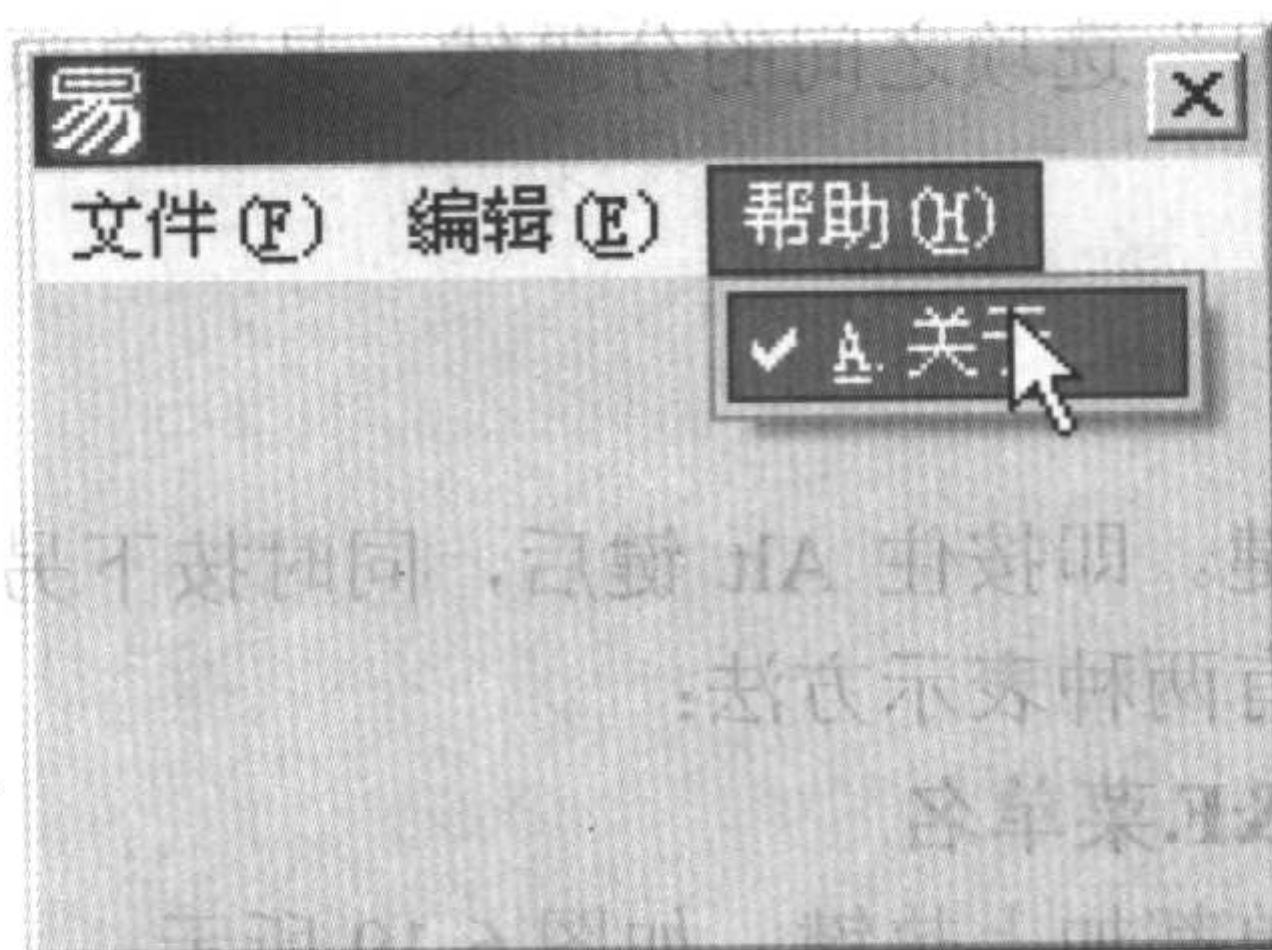


图 6-21 菜单的“选中”属性

## (2) “允许操作”属性

菜单的“允许操作”属性类似按钮组件的“禁止”属性，当菜单的“允许操作”属性被取消选择后，菜单就变成灰色，将不能对该菜单进行任何操作。如图 6-22 所示，图中的“文件”菜单的“允许操作”属性没有选择。

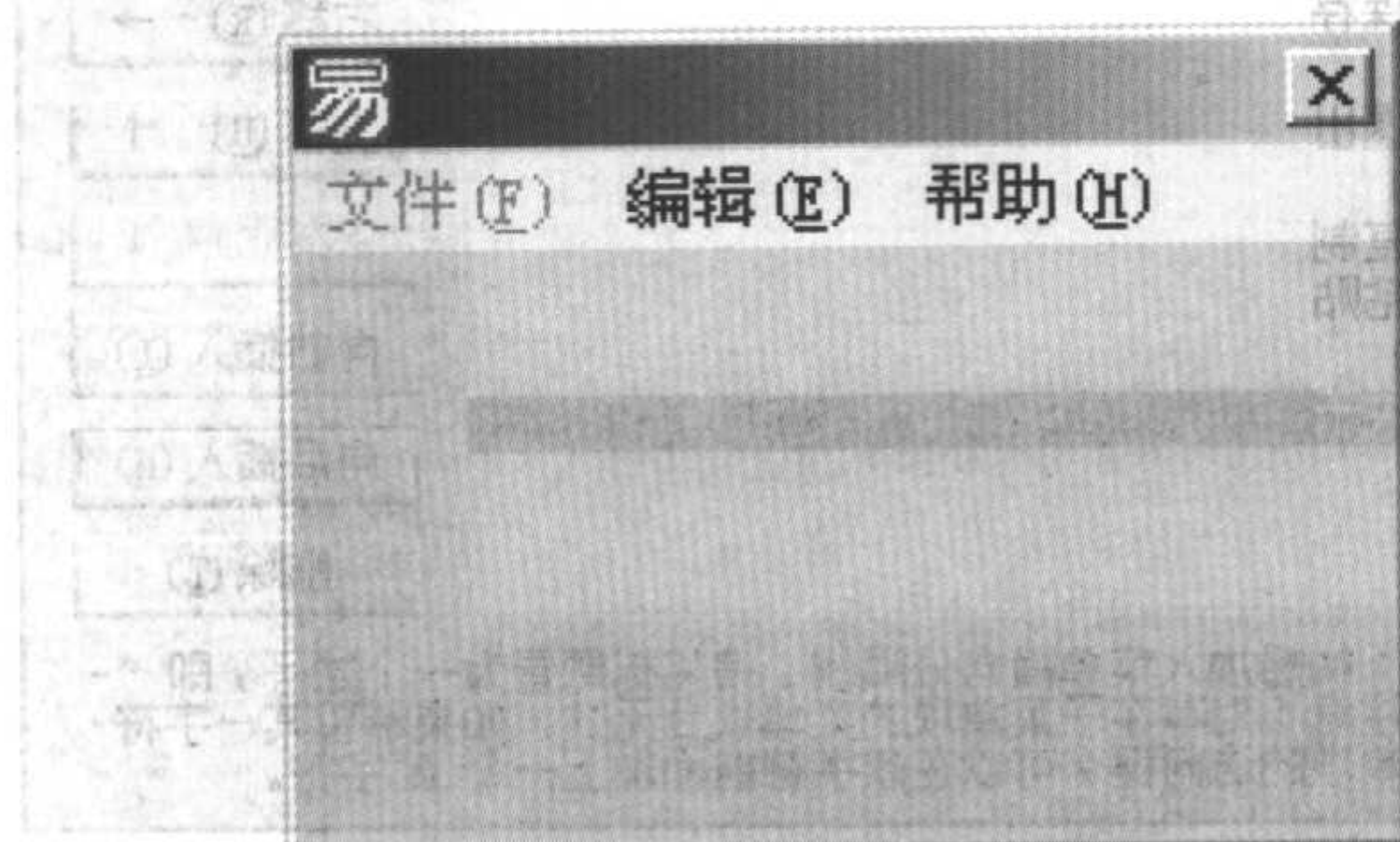


图 6-22 菜单的“允许操作”属性

## (3) “可视”属性

当菜单的“可视”属性被取消选择后，该菜单项就会隐藏，程序运行过程中不可见。如图 6-23 所示，图中的“编辑”菜单被取消“可视”，所以运行后看不到该菜单项。

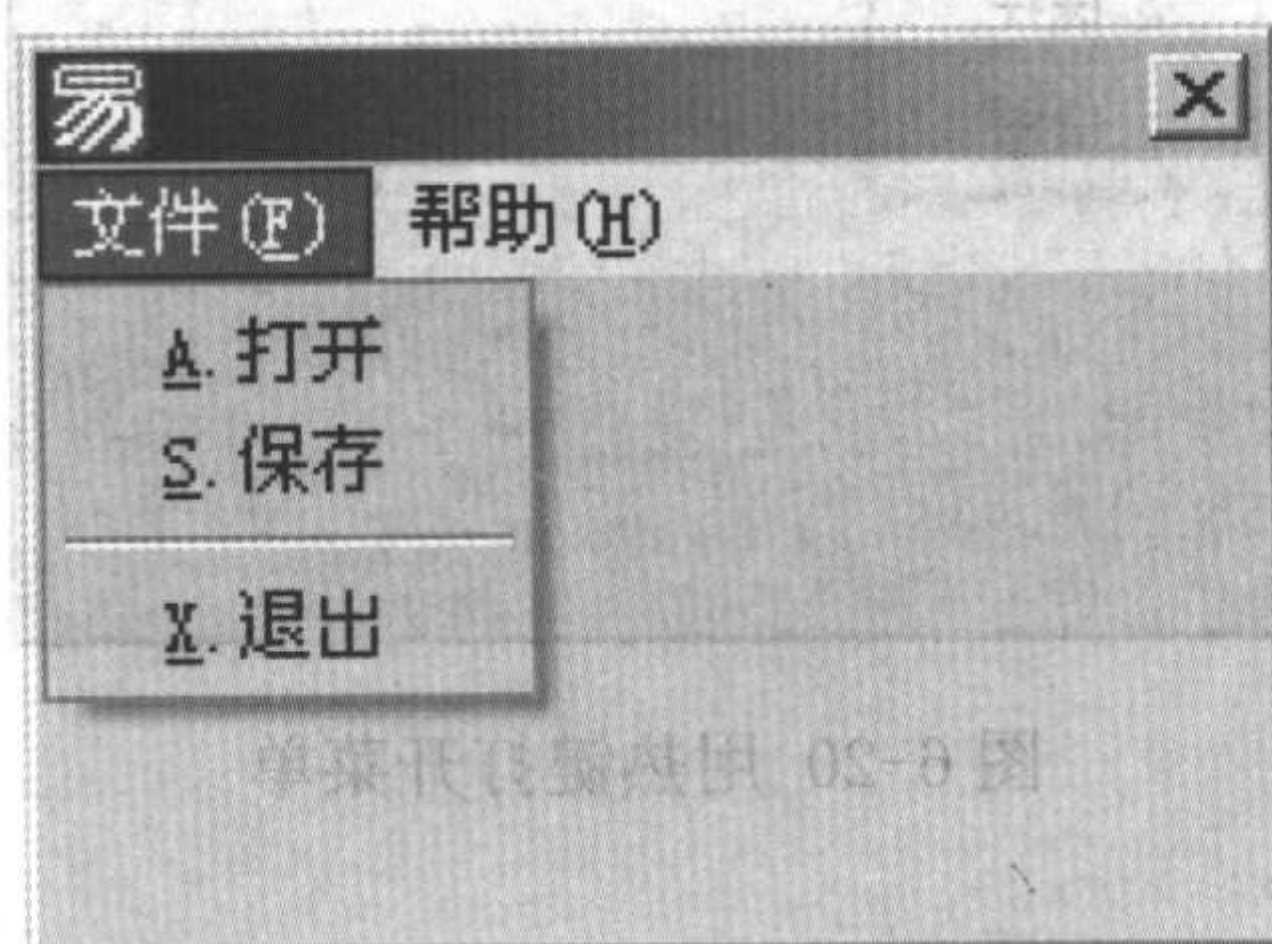


图 6-23 菜单的“可视”属性

## (4) 菜单被选择事件

要实现菜单上各选项的功能，就需要在该项菜单被选择事件子程序中加入相应的代码。在窗口中点击菜单中的一个选项，就会自动生成该项被选择事件子程序。



例如，给刚才编辑过的菜单中的“退出”选项，加入结束程序运行的功能。需要在菜单中选择“退出”选项，然后在生成的“\_退出\_被选择”子程序中输入代码：

子程序名	返回值类型	公开	备注
_退出_被选择			

结束 ()

#### (5) 使用代码控制菜单属性

菜单的属性可以在程序中用代码进行控制，例如：将“编辑”菜单的“可视”属性设置为假：

编辑. 可视 = 假

还可以通过改变菜单的“选中”属性，让该菜单项选中和取消选中。例如，刚才编辑过的菜单，点击菜单中的“关于”选项，在“\_关于\_被选择”子程序中输入代码：

子程序名	返回值类型	公开	备注
_关于_被选择			

```

-- 如果 (关于.选中 = 真)
-- 关于.选中 = 假
-- 关于.选中 = 真

```

当程序运行后，就可以显示和取消“关于”前的“√”。

上述例程参见随书光盘本章目录中的“菜单编辑.e”。

### 6.2.3 弹出菜单

如何实现在程序中单击鼠标右键就弹出一个菜单的功能呢？可以使用“弹出菜单 ()”命令，在指定的位置弹出一个菜单。

“弹出菜单 ()”命令有三个参数，第一个参数填写欲弹出的菜单名，后两个参数控制弹出菜单的位置，可以在任意位置弹出指定菜单。例如，要在鼠标当前位置弹出“编辑”菜单，可以使用代码：

弹出菜单 (编辑, 取鼠标水平位置 (), 取鼠标垂直位置 ())

下面编写一个在图片框中点击鼠标右键就弹出一个菜单的程序。

新建一个易程序，在窗口中添加一个图片框组件，将图片框组件的“边框”属性设置为“镜框式”。

新建一个菜单，名为“图片框菜单”，为其加入2个子菜单“加入图片”和“删除图片”。如图 6-24 所示。并将“图片框菜单”的可视属性设置为假。



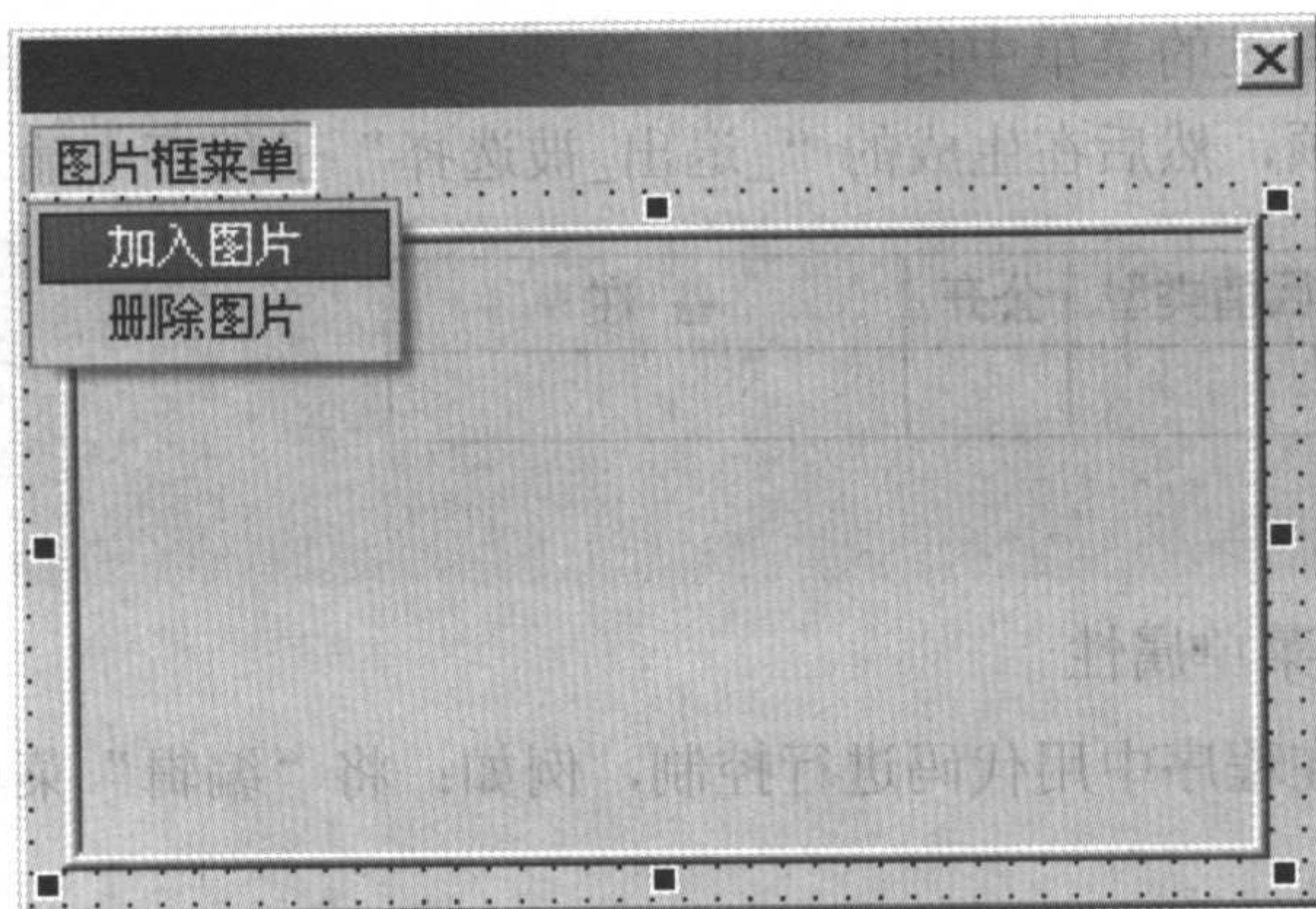


图 6-24 图片框弹出菜单

选中图片框组件，在事件列表中选择“鼠标右键被按下”事件，在产生的“\_图片框1\_鼠标右键被按下”子程序中输入代码：

子程序名	返回值类型	公开	备注		
_图片框1_鼠标右键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

弹出菜单 (图片框菜单, , )

最后，试运行程序，在图片框中点击鼠标右键，可以看到在鼠标位置弹出一个菜单，如图 6-25 所示。



图 6-25 图片框中弹出菜单

要实现通过右键菜单为图片框更换图片的功能，还需要在各菜单项中添加其他代码，请参见随书光盘中的例程“图片框弹出菜单.e”。



## 6.3 对话框

程序中最常用的对话框是“信息框”和“输入框”，“信息框（）”和“输入框（）”都是标准命令。命令运行后会弹出一个固定格式的窗口，根据参数数据变化而显示不同的内容，“信息框”用来输出信息，“输入框”用来输入信息。

### 6.3.1 “信息框（）”命令

“信息框（）”命令的调用格式：

〈整数型〉 信息框（提示信息，按钮，[窗口标题]）

参数 1 的名称为“提示信息”，弹出窗口中所显示的提示信息。

参数 2 的名称为“按钮”，类型为“整数型（int）”，初始值为“0”。参数值由以下几组常量值组成，在将这些常量值相加以生成参数值时，每组值只能取用一个数字（第五组除外）：

第一组（描述对话框中显示按钮的类型与数目）：

0、#确认钮；1、#确认取消钮；2、#放弃重试忽略钮；3、#取消是否钮；4、#是否钮；5、#重试取消钮

第二组（描述图标样式）：

16、#错误图标；32、#询问图标；48、#警告图标；64、#信息图标

第三组（说明哪一个按钮是缺省默认值）：

0、#默认按钮一；256、#默认按钮二；512、#默认按钮三；768、#默认按钮四

第四组（决定如何等待消息框结束）：

0、#程序等待；4096、#系统等待

第五组（其他）：

65536、#位于前台；524288、#文本右对齐

参数 3 的名称为“窗口标题”，类型为“文本型（text）”，可以被省略。参数值指定显示在对话框标题栏中的文本。

命令的第二个参数可以控制信息框按钮的类型与数目，例如：

信息框（“是否关闭？”，#是否钮，“是否关闭”）

命令运行后弹出的信息框，如图 6-26 所示。

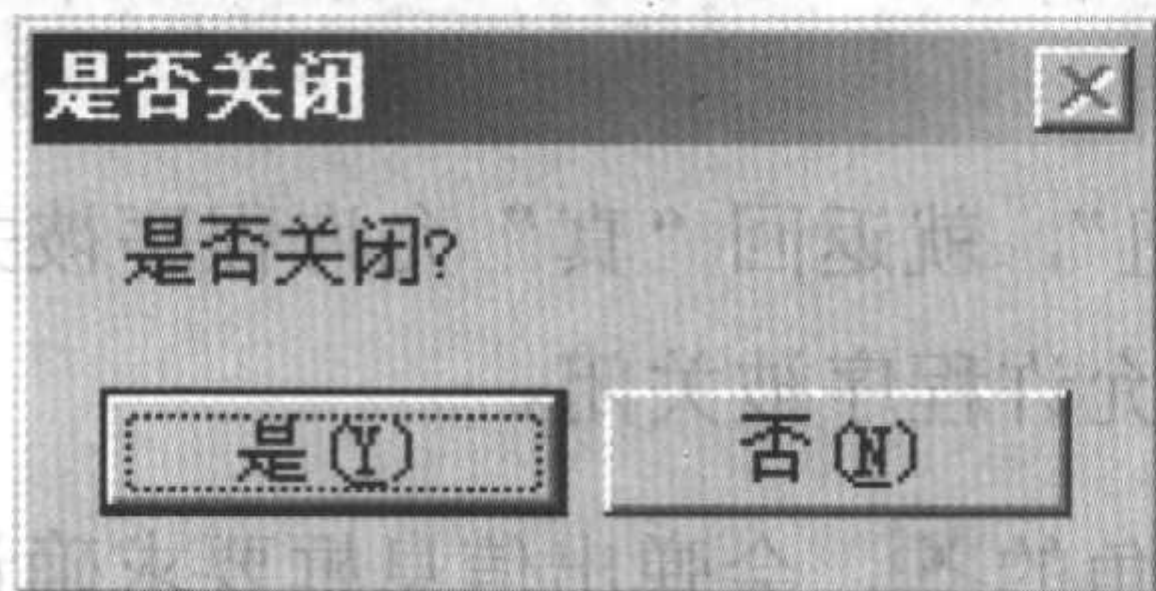


图 6-26 “#是否钮”信息框





命令的第二个参数有五组，除第五组外每组只能选用一个数字，所以，将每组都取一个数字加起来，就可以产生各种效果相加的信息框，例如：

信息框（“出现错误”，#放弃重试忽略钮 + #询问图标 + #默认按钮二 + #系统等待 + #文本右对齐，“错误”）

也可以直接将常量值相加，上面的代码也可以写成：

信息框（“出现错误”，2 + 32 + 256 + 4096 + 524288，“错误”）

该信息框运行后的效果如图 6-27 所示。



图 6-27 信息框第二个参数多个常量相加

## 6.3.2 “信息框 ( )” 命令的返回值

信息框可以显示多个按钮，如何知道用户点击了哪个按钮呢？可以根据信息框的返回值判断用户点击的是那一个按钮。信息框返回值的含义如下：

0、#确认钮； 1、#取消钮； 2、#放弃钮； 3、#重试钮； 4、#忽略钮； 5、#是钮； 6、#否钮。


例如，实现一个程序为：在程序结束前弹出一个有“是否钮”的信息框，如果用户点击了“是”钮，就关闭程序，如果用户点击了“否”钮，程序就不被关闭。

首先，新建一个易程序，然后在“\_启动窗口\_可否被关闭”事件子程序中输入代码：

子程序名	返回值类型	公开	备注
_启动窗口_可否被关闭	逻辑型		

```
如果 (信息框 (“是否关闭程序?”, #是否钮 + #询问图标, “询问窗口”) = #是钮)
    返回 (真)
返回 (假)
```

当信息框返回值为“#是钮”，就返回“真”允许程序被关闭；如果信息框的返回值不是“#是钮”，则返回“假”不允许程序被关闭。

运行程序，点击窗口右上角的 ，会弹出信息框要求确认是否退出。如图 6-28 所示。



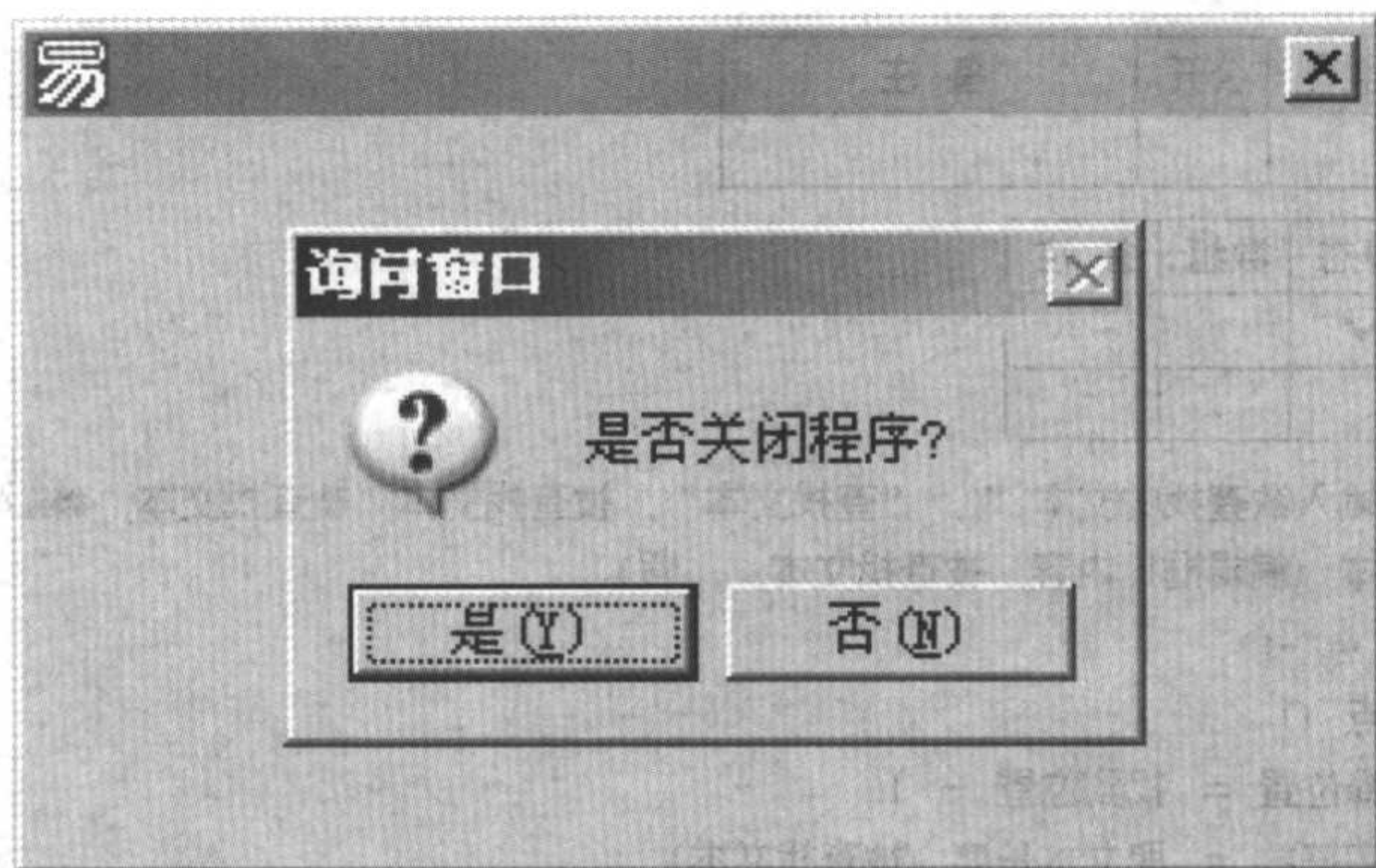


图 6-28 弹出信息框提示“是否关闭窗口”

### 6.3.3 “输入框 ( )” 命令

在程序运行过程中难免需要接受用户输入的数据。如果为接收用户输入的一个简单数据，而单独设计一个接收数据输入的窗口实在没有必要，使用“输入框 ( )”命令就足够了。

输入框相当于动态创建一个固定格式的新窗口，窗口的标题、提示标签的内容以及输入框的初始信息均可以由参数数据动态指定，调用方式如下：

〈逻辑型〉 输入框 ( [提示信息] , [窗口标题] , [初始文本] , 存放输入内容的变量 , [输入方式] )

参数 1 为“提示信息”，类型为“文本型”，参数指定输入框的提示信息。

参数 2 为“窗口标题”，类型为“文本型”，参数值指定显示在对话框标题栏中的文本。如果省略，默认为文本“请输入：”。

参数 3 为“初始文本”，类型为“文本型”，可以被省略。参数值指定初始设置到对话框输入文本框中的内容。

参数 4 为“存放输入内容的变量”，类型为“通用型”，提供参数数据时只能提供变量。参数值所指定的变量可以为数值或文本型，用于以不同的数据类型取回输入内容。

参数 5 为“输入方式”，类型为“整数型”，可以被省略。参数值可以为以下常量值： 1、#输入文本； 2、#输入整数； 3、#输入小数； 4、#输入密码。默认为“#输入文本”。

### 6.3.4 “输入框 ( )” 命令的应用

在程序设计中，接受少量内容输入的情况均可使用“输入框”命令，能够极大的简化代码复杂程度。

例如，在编辑框中查找指定的文本，可以使用输入框来指定待查找的文本，当用户点击了“确认输入”按钮，就开始查找，找到后就自动选中被找到的文本。

首先新建一个易程序，在“\_启动窗口”中添加 1 个编辑框组件和 1 个按钮组件，将按钮组件的标题改为“查找”。

双击按钮，在“\_按钮 1\_被单击”子程序中输入代码：





子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
被查找文本	文本型	✓		
找到位置	整数型			

```
如果真 (输入框 ( "输入欲查找的文本:" , "查找文本" , 被查找文本 , 被查找文本 , #输入文本 ) = 真)
    找到位置 = 寻找文本 (编辑框1.内容, 被查找文本, , 假)
    如果 (找到位置 ≠ -1)
        编辑框1.获取焦点 ()
        编辑框1.起始选择位置 = 找到位置 - 1
        编辑框1.被选择字符数 = 取文本长度 (被查找文本)
    信息框 ("没有找到指定文本" , 0 , )
```

上述代码中,对“输入框( )”命令的返回值进行判断。如果用户点击了输入框中的“确认输入”按钮,程序就从第一个位置开始查找输入的文本。找到了目标文本后就将该段文本选中。

最后试运行程序,在编辑框中输入一段文字,然后点击“查找”按钮,在弹出的输入框中输入欲查找的文本,然后点击“确认输入”按钮。如图6-30所示。被找到的文本会被选中。

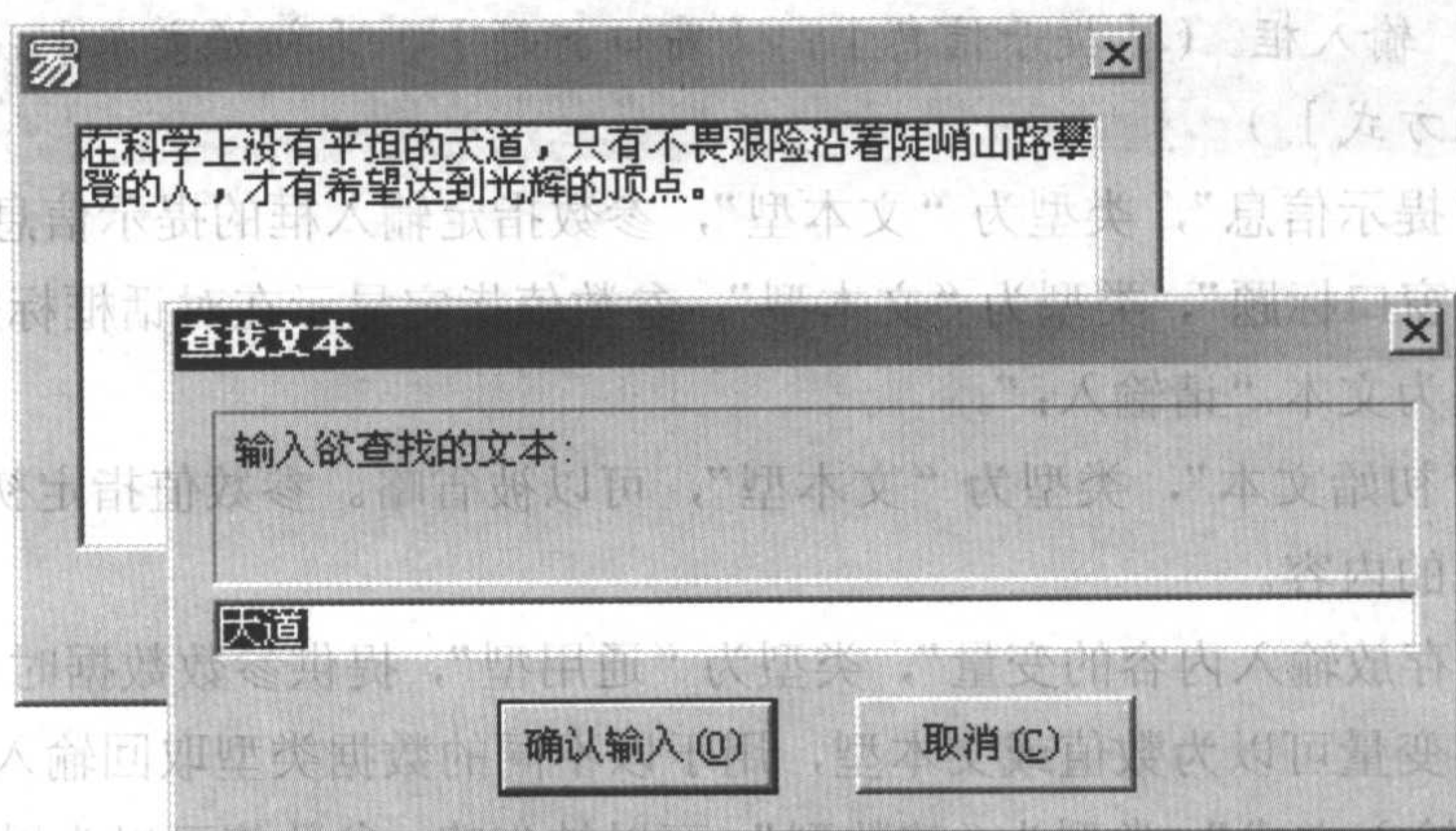


图 6-30 查找指定文本

本例程参见随书光盘中例程“查找文本.e”。

## 6.4 本章小结

针对于当前窗口中的方法可以直接使用方法命令,既可以写成“\_启动窗口.刷新( )”也可以写成“刷新( )”,再例如“\_启动窗口.销毁( )”可以直接写为“销毁( )”。作用是一样的。

大家还可进行以下的练习:



1. 熟悉窗口的属性、事件和方法。为程序新增加窗口，并在启动窗口中用按钮弹出它们。
2. 在窗口的“底图”属性中载入一张图片，学习制作图形窗口界面，可以用截图的方法，仿照其他软件的图形界面制作出一个漂亮的窗口来。
3. 编写一个双语菜单程序，可通过改变两组菜单的显示与否，在中文菜单与英文菜单之间切换；或使用程序代码直接修改菜单的标题。
4. 编写一个进度拷贝文件程序，当拷贝出现问题时弹出“放弃、重试、忽略”按钮的信息框，当点击信息框中的不同按钮，就进行不同的操作。
5. 使用输入框来接受用户输入的数据，并显示到窗口的一个标签中。





## 第七章 组件介绍

### 7.1 易语言组件简介

在易语言程序开发中经常要用到组件，前面章节已经提过，窗口是一种特殊的组件，它可以作为其他组件的载体。这里所说的其他组件，和窗口一样，都是创建界面的基本结构模块。一个 Windows 窗口程序一般都要包含若干个不同的组件，相互组合分别实现不同的功能。如果将开发出的程序形象地比喻成一个产品的话，那么组件就好比是组成这件产品的一个个“零件”。而将这些“零件”组成一件“产品”的过程，其实就是开发者为组件编写代码和集成组件程序的过程。

易语言新版本中，各种常用组件已达到 80 个之多。这些组件涉及软件开发中各方面的应用。随着版本不断升级，组件数量还将不断地增加，限于篇幅，不能对每个组件都进行详细的介绍，只能列举几个常用组件的属性、方法和事件，侧重培养大家对组件使用的自学能力。今后即使遇到没用过的新组件，也能很快地了解该组件的特性并熟练掌握。

#### 7.1.1 易语言内部组件

在新建一个易程序后，可以在程序主界面右边的窗口组件箱中看到当前系统所支持的所有组件，包括基本组件、扩展组件、外部组件和外部事件组件四类。如图 7-1 所示。



图 7-1 易语言组件箱



易语言基本组件是最常用的组件，可按照功能大致分为十类。如表 7-1 所示。


表 7-1 基本组件分类表

基本组件分类名称	基本组件图标和名称	基本组件用途
文字显示组件	 “标签”  “编辑框”	用于输入、显示和编辑单项文本数据
列表类组件	 “组合框”  “列表框”  “选择列表框”	显示多条记录和数据
按钮组件	 “按钮”  “图形按钮”  “单选框”  “多选框”  “超级链接框”	主要用于鼠标点击时执行各种操作
文件系统组件	 “目录框”  “驱动器框”  “文件框”	主要用于对计算机硬盘中的文件进行管理
图形处理和多媒体组件	 “画板”  “图片框”  “颜色选择器”  “外形框”  “影像框”	用于对图形文件和多媒体文件的显示和处理
网络组件	 “数据报”  “服务器”  “客户”  “端口”	主要用于网络之间的通讯
滚动组件	 “横向滚动条”  “纵向滚动条”  “调节器”  “进度条”  “滑块条”	主要用于不能自动提供滚动条的组件
时间组件	 “日期框”  “月历”  “时钟”	用于显示时间和日期





表 7-1 基本组件分类表

	 “表格”  “数据源”	
数据库组件	 “通用提供者”  “数据库提供者”	用于处理内部和外部数据库数据
	 “外部数据库”  “外部数据提供者”	
其他类别组件	 “分组框”  “通用对话框”	
	 “选择夹”  “打印机”	

常用标准扩展组件分类如表 7-2 所示。

表 7-2 扩展组件分类表

扩展组件分类名称	扩展组件图标和名称	扩展组件用途
文字显示组件	 “透明标签”  “超级编辑框”	用于输入、显示和编辑单项文本数据
列表类组件	 “树形框”  “超级列表框”	显示多条记录和数据
按钮组件	 “工具条”	主要用于鼠标单击时执行各种操作
多媒体组件	 “高级影像框”	用于多媒体文件的显示和处理
网络组件	 “超文本浏览框”  “IP 编辑框”	主要用于网络之间的通讯
数据库组件	 “数据库连接”  “记录集”	用于处理内部和外部数据库数据
其他类别组件	 “分隔条”  “状态条”	

除基本组件和扩展组件以外，还有两个分类是：外部组件和外部事件组件。外部组件是由注册到易语言中的 OCX 或类型库自动增加的，外部事件组件是注册的类型库对象所拥有的组件事件。这两类组件的具体使用方法将在后续章节中单独介绍。



组件分类还可以根据其可否容纳其他组件划分为容器类和非容器类，或按运行时是否具有可视外形划分为界面类和功能类。容器类组件内可以包容其他的组件，如选择夹、图片框、外形框等组件；而功能类组件仅用作提供某种功能，运行时不可见，如时钟、打印机等组件。

易语言为程序开发人员所提供的组件已相当的全面，只要在编程中合理地使用这些组件，灵活搭配，就能使程序拥有漂亮的界面和强大的功能，并且能提高程序开发的效率。

大部分组件都拥有自己的属性、方法和事件，也就是说组件是属性、方法和事件的集合。组件也被称为“对象”，比如说，一个人、一件东西、一件事情，都可以被认为是一个“对象”。对象的属性记录着对象的特征，对象的方法提供了操作对象的途径，对象的事件用作通知外部其状态发生了改变。譬如说一个电源开关，其外形、颜色、使用电压等等都可以认为是该电源开关“对象”的属性，而关闭或打开此电源开关则可以认为是电源开关“对象”的方法。在关闭或打开的同时，它可能产生事件，如通知与其相连的电器开始工作或停止等等。在易程序中，如“画板”组件，即是一个典型的对象，它具有“画笔类型”、“画笔粗细”等属性，“画直线”、“画矩形”等方法，在被重新绘画时还会产生“绘画”事件。

## 7.2 组件的属性

组件的属性用来描述此对象的状态。“名称”属性设置组件的名称标识，在程序中必须使用此标识引用该组件，因此，组件的名称标识不可重复。在程序窗体中添加组件，组件名自动设为“组件类型+序号”，如添加第一个按钮组件，其名称自动设为“按钮 1”，添加第二个按钮组件，其名称自动设为“按钮 2”，以此类推。为了增加程序的可读性，建议修改初始名称为有实际意义的名称标识，便于理解和记忆。

### 7.2.1 组件的共有属性

组件共有属性是组件的基本属性，是每个可视组件都必需具有的，包括“左边”、“顶边”、“宽度”、“高度”、“标记”、“可视”、“禁止”和“鼠标指针”等。

1. “左边”、“顶边”是指当前组件相对于其所在容器组件的左边和顶边的位置。如图 7-2 所示。

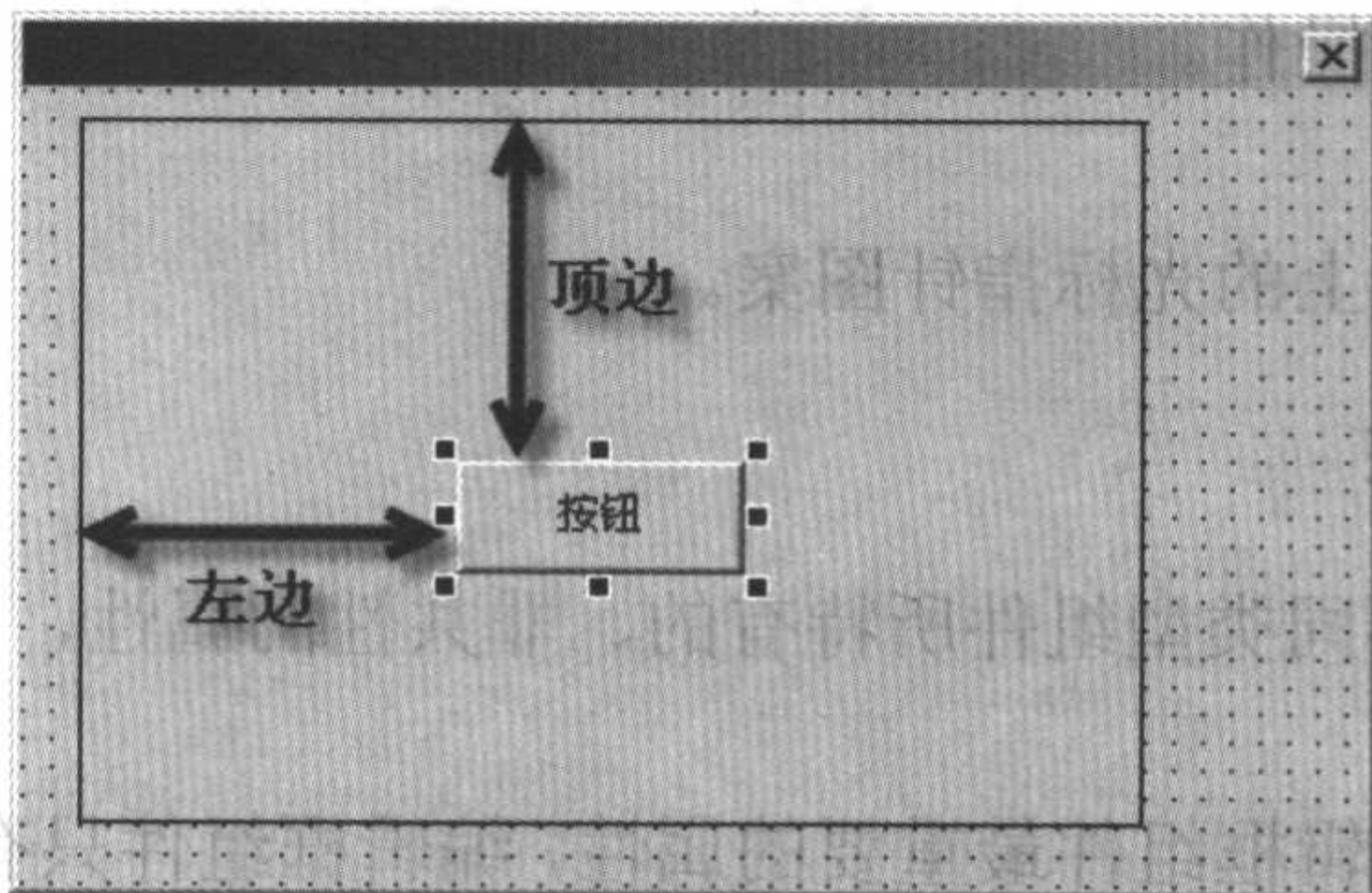


图 7-2 “左边”“顶边”属性





**注意：**欲将组件添加到容器组件中，应先选中容器组件，再在组件箱中选择被添加组件，然后单击容器组件，即可将组件添加至容器组件中。“工具条”组件的添加方法有点特殊，首先将“工具条”组件添加到设计窗体，剪切“工具条”组件，然后选中容器组件，粘贴。

## 2. “宽度”与“高度”属性

设置指定组件的宽度与高度。当超出组件所在容器范围后，超出部分不可见。

## 3. “标记”属性

通常设置为数字，可以用“取标记组件( )”取出对应的组件，但在一个判断中只能取同一类型的组件。

“标记”的其中一种使用方法：

在设计窗体上添加五个“按钮”组件，分别设置其“标记”属性为 1、2、3、4、5。

程序代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
序号	整数型			
按钮变量	按钮			

--- 计次循环首 (4, 序号)

按钮变量 = 取标记组件 (序号)

事件转移 (按钮变量, 按钮5)

--- 计次循环尾 0

子程序名	返回值类型	公开	备注
__按钮5_被单击			

信息框 (“按钮5”, 0, )

运行程序，单击不同按钮，都会弹出信息框。

## 4. “可视”和“禁止”属性

“可视”属性是控制组件在程序运行时是否在窗体上显现。“禁止”属性就是禁止用户使用，虽然可见但不能被操作。

## 5. “鼠标指针”属性

设置鼠标移动到组件上的光标指针图案。

## 7.2.2 组件的专有属性

组件专有属性是指不同类型组件所特有的、非共性的属性，下面以编辑框组件为例进行说明：

1. “内容”属性是编辑框组件最重要的属性，编辑框组件只有“内容”属性，没有“标题”属性。



“内容”属性存放编辑框中的文本。“内容”与“标题”属性的不同在于，“内容”属性的值允许在程序运行中由使用者直接进行修改，而“标题”属性的值必须通过程序代码进行修改。

打开随书光盘本章目录中的例程“编辑框-内容属性.e”。本例中用三个按钮控制一个编辑框中的内容。如图 7-3 所示。

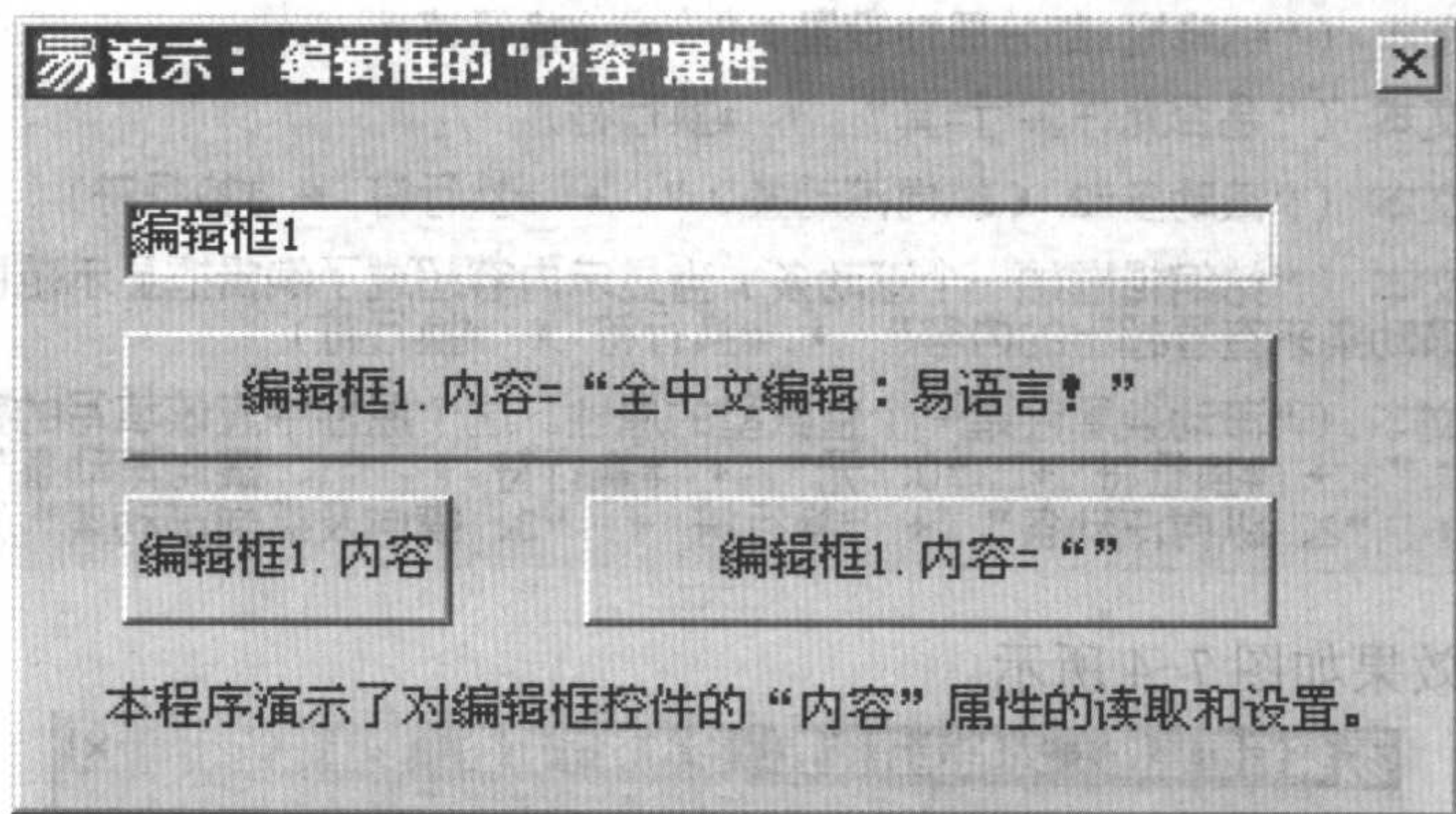


图 7-3 运行“编辑框-内容属性.e”例程

“\_按钮 1\_被单击”事件子程序改变编辑框中的内容，程序代码如下：

```
编辑框 1. 内容 = “全中文编辑：易语言！”
```

“\_按钮 2\_被单击”事件子程序用信息框显示编辑框中的内容，程序代码如下：

```
信息框 (编辑框 1. 内容, 0, )
```

“\_按钮 3\_被单击”事件子程序清空编辑框中的内容，程序代码如下：

```
编辑框 1. 内容 = “”
```

## 2. “起始选择位置”属性

当编辑框中的文本有一部分处于“反选”状态时，“起始选择位置”属性描述了被选择文本的起始位置（从 0 开始）；如果编辑框中没有文本被选中，则“起始选择位置”指出光标所在位置（从 0 开始）。

## 3. “被选择字符数”属性

返回或设置所选择的字符数。如果设置字符数时使用值 -1，则选择文本框内的所有字符。

## 4. “被选择文本”属性

返回或替换当前所选择的文本。

默认情况下，“起始选择位置”、“被选择字符数”的值均为 0，“被选择文本”的值为空文本（即“”）。

如果程序代码为“起始选择位置 = -1”，则将当前光标位置移动到文本尾部；如果令“被选择字符数 = -1”，则选择编辑框内所有的字符；如果在代码中向“被选择文本”属性赋值，则将编辑框中原来被选中的字符替换为新的字符串。（注意：“被选择文本”属性只能在运行时设置，设计时不可改动。）

打开随书光盘中的例程：“编辑框-选择类属性.e”。





“容”例程在窗口设计阶段并未在编辑框中写入任何文本，而是由“\_启动窗口\_创建完毕”事件子程序在窗口建立完毕时将文本写入到编辑框中。程序代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

- 编辑框1.加入文本 (“编辑框1部分属性设置:” + #换行符)
- 编辑框1.加入文本 (“是否允许多行=真” + #换行符)
- 编辑框1.加入文本 (“滚动条=2 (纵向滚动条)” + #换行符 + #换行符)
- 编辑框1.加入文本 (“给编辑框加一个滚动条，当显示内容超过了编辑框显示范围后，可以拖动滚动条来查看超出的内容” + #换行符 + #换行符)
- 编辑框1.加入文本 (“滚动条属性是一个整数型的属性。这个属性中应该填写的整数代表  
的意义:” + #换行符 + “0、无” + #换行符 + “1、横向滚动条” + #换行符 + “2、纵向滚动条” + #换行符 + “3、横向及纵向滚动条”)

运行程序，效果如图 7-4 所示。

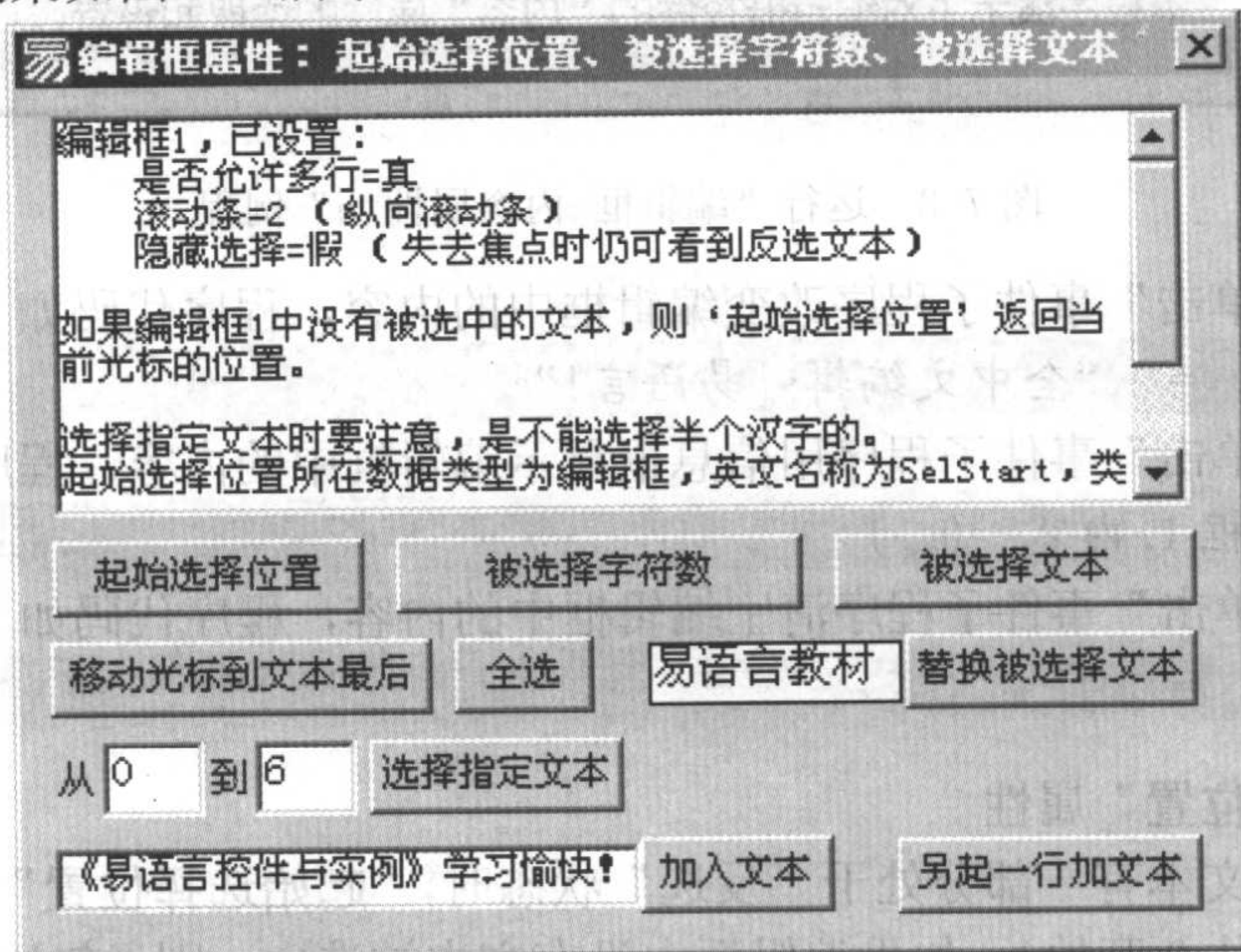


图 7-4 运行“编辑框-选择类属性.e”例程

“\_按钮 1\_被单击”事件子程序设置“起始选择位置”属性。运行时，任意用鼠标点击编辑框某一个位置后，再点击此按钮就会弹出信息框，显示鼠标目前所在位置为第几个字符。如果是选中一些文字，点击这个按钮后，就会显示起始选择的位置。程序代码如下：

信息框 (“起始选择位置:” + 到文本 (编辑框 1. 起始选择位置), 0, )

“\_按钮 2\_被单击”事件子程序设置“被选择字符数”属性，如果选中一些文字，点击这个按钮，就会显示被选择的字符数。如果没有选择字符，将显示为“0”。程序代码如下：

信息框 (“被选择字符数:” + 到文本 (编辑框 1. 被选择字符数), 0, )

“\_按钮 3\_被单击”事件子程序设置“被选择文本”属性，如果选中了一些文字，点击这个按钮，就会显示所选择的文本。如果没有选择字符，那么将显示空白文本。程序代码如下：

信息框 (编辑框 1. 被选择文本, 0, )



以上三个按钮被单击事件中，代码后面都有“编辑框 1. 获取焦点( )”方法，这是因为单击按钮时，按钮获得焦点，编辑框同时失去焦点。只有恢复编辑框的焦点，才可以看到编辑框中的光标。

“按钮 6”是对“按钮 1”的实际应用。代码如下：

编辑框 1. 起始选择位置 = -1

“-1”值是编辑框组件所规定的。即：如果使用值-1，则将当前光标位置移动到文本尾部。大家可以在设计程序界面时，鼠标指向相应属性并按 F1 键，就会显示即时帮助。

“按钮 5”的功能是全选编辑框中的文字。“\_按钮 5\_被单击”代码如下：

编辑框 1. 被选择字符数 = -1

“-1”值是“被选择字符数”属性所规定的。即：如果设置字符数时使用“-1”值，则选择文本框内的所有字符。

“按钮 4”是对“按钮 3”的实际应用。“\_按钮 4\_被单击”事件子程序代码如下：

编辑框 1. 被选择文本 = 编辑框 2. 内容

这行代码直接将编辑框 2 中的内容替换为编辑框 1 中被选择的文本，以达到替换被选择文本的目的。

“\_按钮 7\_被单击”事件子程序代码如下：

子程序名	返回值类型	公开	备注
_按钮7_被单击			

```

--- 如果真 (到数值 (编辑框4. 内容) > 到数值 (编辑框3. 内容))
    编辑框1. 起始选择位置 = 到数值 (编辑框3. 内容)
    编辑框1. 被选择字符数 = 到数值 (编辑框4. 内容) - 到数值 (编辑框3. 内容) + 1
编辑框1. 获取焦点 ()
    
```

第一行代码是对两个编辑框内输入的数值进行比较，如果条件成立，将执行“如果真”判断语句内的代码，而这两行代码即可以根据“编辑框 3”的内容确定“编辑框 1”中起始选择位置，根据“编辑框 4”的内容确定“编辑框 1”中被选择的字符数，用这两行命令，就可实现选中“编辑框 1”中的一段文字。

“按钮 8”被单击事件程序代码如下：

编辑框 1. 加入文本 (编辑框 5. 内容)

此行代码使用“加入文本( )”方法，将编辑框 5 中的内容直接加入到编辑框 1 内容的后面。

使用“按钮 10”可将文本另起一行加入到编辑框 1 中。程序代码如下：

编辑框 1. 加入文本 (#换行符 + 编辑框 5. 内容)

在内容前加了一个“#换行符”，以便与前一行加以区分。

打开随书光盘中的例程：“编辑框-类似输入框.e”。

可以看到在“\_启动窗口\_创建完毕”事件子程序代码如下：





子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

编辑框1. 获取焦点 0

编辑框1. 内容 = “请重新输入口令”

编辑框1. 起始选择位置 = 0

编辑框1. 被选择字符数 = 14

程序运行后如图 7-5 所示。



图 7-5 运行“编辑框-类似输入框.e”例程

可以看到，编辑框中的内容全部被选中，如果这时输入新的口令，那么将会自动去除原编辑框中的内容。

要注意第一行代码“编辑框 1. 获取焦点( )”一定要有，如果没有此行代码，在运行时，光标将无法定位在编辑框中，也就无法反选编辑框中的内容了。

点击名称为：“类似输入框”的按钮，可以看到，这里事先选中初始文本作为提示。如图 7-6 所示。

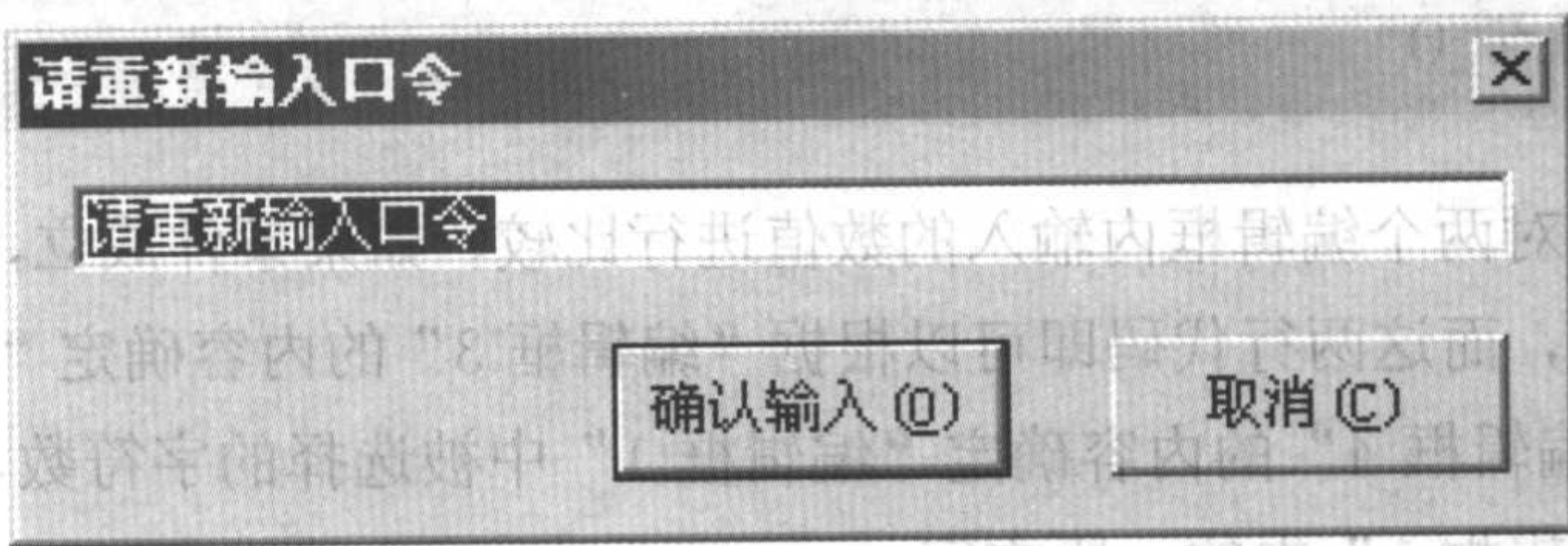


图 7-6 “易语言”输入框

程序代码如下：

子程序名	返回值类型	公开	备注
__按钮_被单击			

变量名	类型	静态	数组	备注
临时变量	文本型			

输入框 ( “请重新输入口令”, “请重新输入口令”, 临时变量, )

## 5. “是否允许多行”属性

逻辑型，只能为“真”或“假”，默认为“假”。当本属性为“假”时，编辑框中只能



输入一行文本。如果想要编辑框容纳多行文本，就需要把该属性改为“真”。可直接修改编辑框属性，或使用程序命令：

编辑框.是否允许多行 = 真

## 6. “滚动条”属性

整数型，有以下4个可选值：“0.无”、“1.横向滚动条”、“2.纵向滚动条”、“3.横向及纵向滚动条”。默认为“0.无”。只有当编辑框的“是否允许多行”属性为“真”时，本属性才可操作。

在多行编辑框中（即“是否允许多行”属性为真），一般情况下把“滚动条”属性设为“2.纵向滚动条”。

打开随书光盘中的例程：“编辑框-多行与滚动条.e”。这个例程同时演示了“是否允许多行”与“滚动条”两个属性。如图7-7所示。

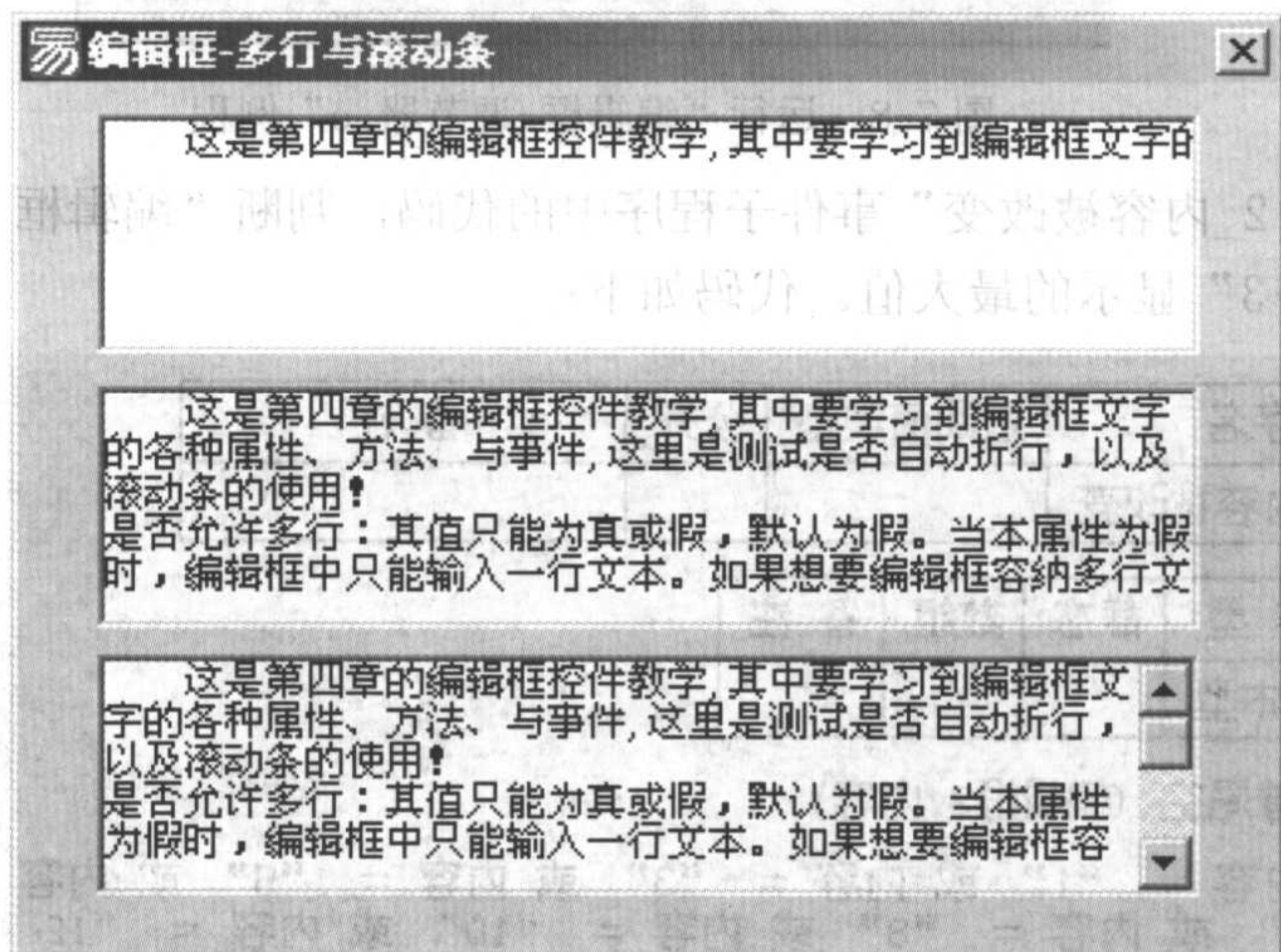


图 7-7 运行“编辑框-多行与滚动条.e”例程

其中上方的编辑框“是否允许多行”属性为“假”，没有滚动条。后面的内容都被截断，无法看见。

中间的编辑框“是否允许多行”属性为“真”，也没有滚动条，如果要放更多的内容就要用鼠标向下拖动，或用光标一行一行向下移，非常不方便。

下方的编辑框，“是否允许多行”属性为“真”，“滚动条”的属性为“2”，显示一个纵向滚动条，这样就可以通过滚动条来翻看更多的内容了。

## 7. “调节器方式”属性

整数型，有三个可选值：“0.无调节器”、“1.自动调节器”、“2.手动调节器”。默认值为“0.无调节器”。本属性只有在单行方式（即“是否允许多行”属性为“假”）时才可操作。

调节器通常设置为“1.自动调节器”，这样，单击上下箭头，编辑框中的数值会自动加减1，而且还可以用另外两个属性“调节器底限值”（默认为0）、“调节器上限值”（默认为100）来控制数字变化范围。如果自动调节器不能满足要求，可以采用手动调节器（置“调节器方式”属性为2），这时需要响应编辑框的“调节钮被按下”事件。





打开随书光盘中的例程：“编辑框-调节器.e”。如图 7-8 所示。  
在这里，三个编辑框的调节器都设置为自动调节，因此，每点击一下调节器，增减数为 1。

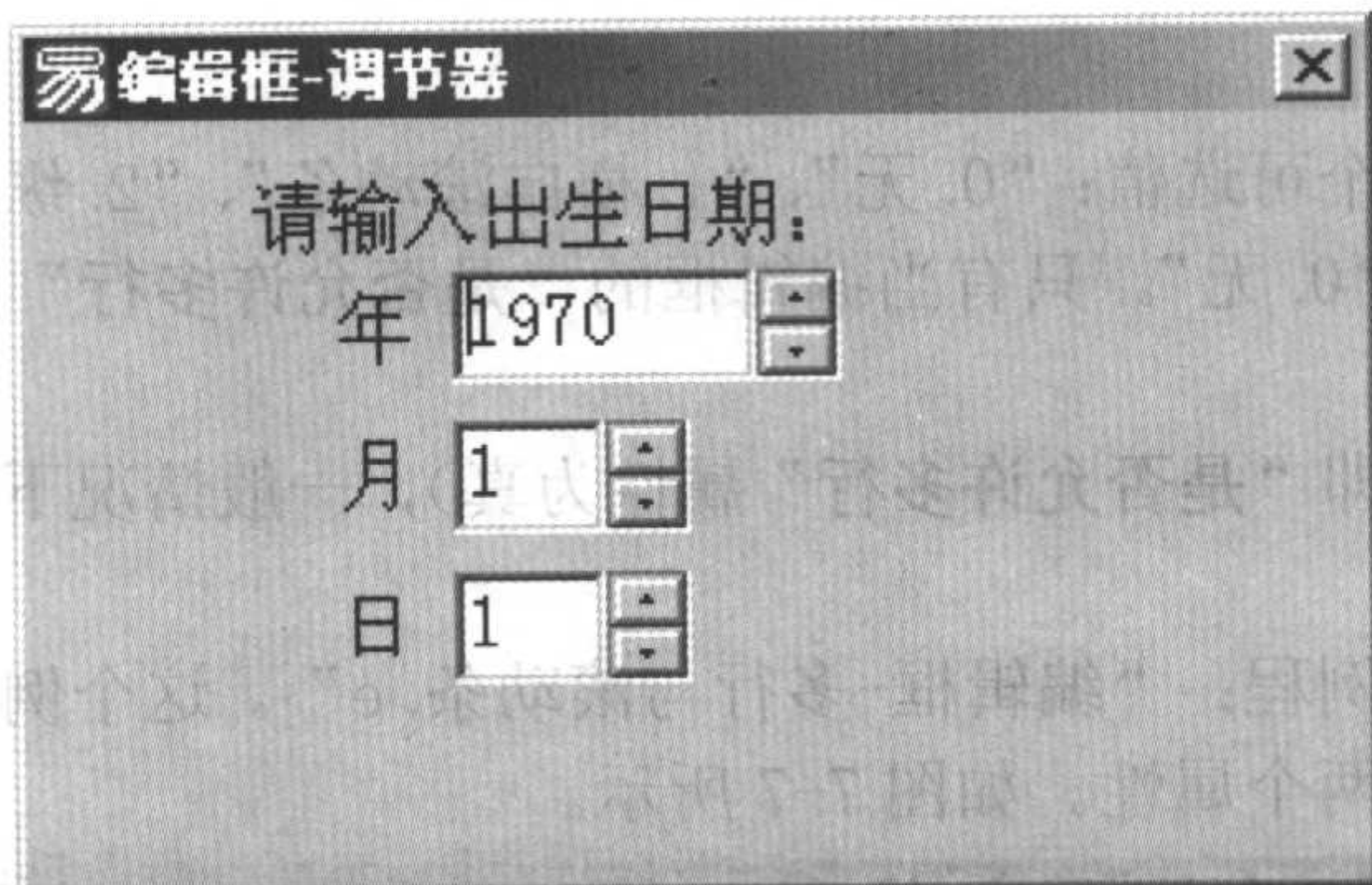


图 7-8 运行“编辑框-调节器.e”例程

“\_编辑框 2\_内容被改变”事件子程序中的代码，判断“编辑框 2”中显示的月份，以控制“编辑框 3”显示的最大值。代码如下：

子程序名	返回值类型	公开	备注
_编辑框2_内容被改变			

变量名	类型	静态	数组	备注
内容	文本型			

内容 = 删首尾空 (编辑框2.内容)

判断 (内容 = “1” 或 内容 = “3” 或 内容 = “5” 或 内容 = “7” 或 内容 = “8” 或 内容 = “10” 或 内容 = “12” )

编辑框3.调节器上限值 = 31

判断 (内容 = “4” 或 内容 = “6” 或 内容 = “9” 或 内容 = “11” )

编辑框3.调节器上限值 = 30

判断 (内容 = “2” )

编辑框3.调节器上限值 = 28

注意：上述程序中“删首尾空 ()”是系统提供的命令，可以将文本前的空格与文本后的空格删除。本例未对闰月做处理，通过时间日期类命令进行判断会得到准确结果。

## 8. “输入方式”属性

整数型，有以下可选值：“0. 通常方式、1. 只读方式、2. 密码输入、3. 整数文本输入、4. 小数文本输入、5. 输入字节、6. 输入短整数、7. 输入整数、8. 输入长整数、9. 输入小数、10. 输入双精度小数、11. 输入日期时间”。默认值为“0. 通常方式”。

当“输入方式”为 1 时（只读方式），程序使用者无法编辑编辑框中的内容；



当“输入方式”为 2 时（密码输入），输入编辑框中的字符将以“\*”的形式显示（显示符号由编辑框的另一个属性**密码遮盖字符**控制）；

当“输入方式”为 3 时，只有数字字符(0 1 2 3……9)和负号(“-”)才可以输入到编辑框中。

书中例程：“编辑框-输入方式.e”演示了所有输入方式运行后的效果。大家可以运行体会一下。

#### 9. “转换方式”属性

整数型，有以下可选值：“0. 无”、“1. 大写->小写”、“2. 小写->大写”。默认为“0. 无”，不进行输入转换。

如果“转换方式”为 1，则输入到编辑框中的字母全都显示为小写；

如果“转换方式”为 2，则输入到编辑框中的字母全都显示为大写。所有输入转换只涉及字母，非字母字符原样显示，不进行任何转换。

书中例程：“编辑框-转换方式.e”演示了所有转换方式。可以输入一段英文或拼音，看看转换效果。

#### 10. “最大允许长度”属性

整数型，控制编辑框中所能容纳的最大字符数。默认为 0，表示不限制。

#### 11. “隐藏选择”属性

逻辑型，只能为“真”或“假”，默认为“真”。如果本属性设置为“假”，即使编辑框失去输入焦点，原来反选的文字仍然反选。

例如：在一个窗口中有一个编辑框和一个按钮组件，当运行时，在编辑框中写入文字，并选中其中一段文字，使用 TAB 键或鼠标单击按钮将焦点移到按钮组件，如果此属性为“假”，那么编辑框中原来选择的反白显示还可看到，如果此属性为“真”，那么原来选择的反白显示不能看到。

#### 12. “文本颜色”、“背景颜色”、“字体”属性

分别设置编辑框的文本颜色、背景颜色和字体。

#### 13. “数据源”、“数据列”属性

与数据库操作相关，将在相关章节中介绍。

### 7.2.3 动态修改组件属性

在属性面板中设置属性，都是手动去修改相应的属性值，这些修改工作必须在运行程序前就完成，但在程序运行期间想调整某个属性，就必须动态的修改组件属性。下面就介绍一下如何动态的修改组件的属性。

动态改变组件属性的基本方法是：

**组件名. 组件属性=组件属性值**

新建易程序。添加一个“画板”组件和两个“按钮”组件，将“画板”组件的“边框”属性设置为“单线边框式”。在组件“按钮 1”的“\_按钮 1\_被单击”事件子程序中添加代



码。程序代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```

画板1.自动重画 = 真
画板1.边框 = 2
画板1.画笔粗细 = 10
画板1.画笔颜色 = #绿色
画板1.刷子颜色 = #蓝色
画板1.画矩形 (20, 20, 画板1.宽度 - 40, 画板1.高度 - 40)
    
```

按“F5 键”运行程序，效果如图 7-9 所示。

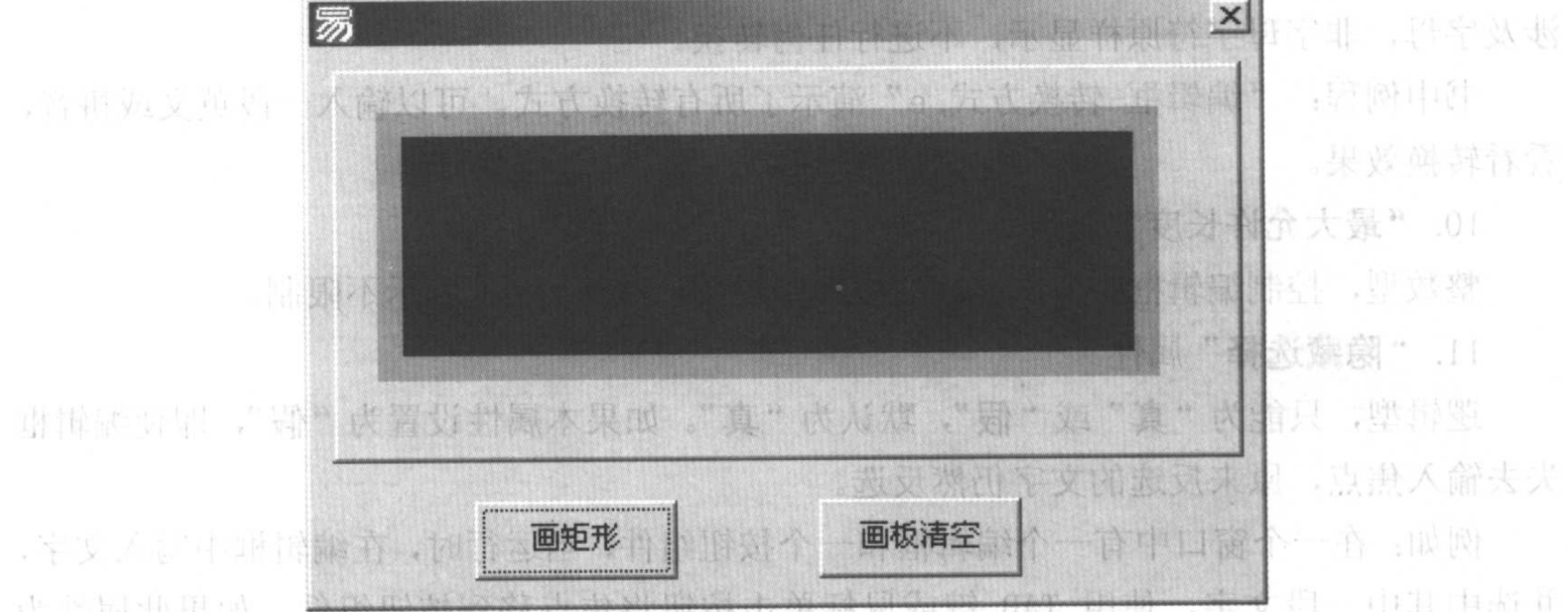


图 7-9 动态设置组件属性程序效果

**注意：**在给组件动态改变属性的时候，一定要注意属性本身的数据类型。也就是说，文本型的组件属性需要赋值文本型的数据，整数型的组件属性就需要赋值整数型的数据。如：

标签 1.标题 = “动态改变标签属性”

上述程序中，标签的“标题”属性是文本型的，所以所赋的值需要用双引号括起来，表示是文本型的数据。即代码中的“动态改变标签属性”。

标签 1.边框 = 6

上述程序中，标签的边框属性是整数型的，所以改变边框属性时就要给边框属性赋一个整数型的数据，即代码中的 6。

画板 1.底图 = { }

上述程序中，画板的底图属性为字节集型，用“{ }”设置为空字节集，画板底图被清空。



## 7.3 组件的专有方法

组件的方法和命令的用法相同，格式是：

组件名.方法名(参数)

组件的方法会对相应的组件产生不同的效果。譬如“画板 1.画矩形(,,,)”就是画板的方法之一。

下面用画板组件制作一个九九表。

新建一个易程序。添加一个“画板”组件和一个“标签”组件。“画板”组件的“边框”属性设置为“镜框式”，“标签”组件的“边框”属性设置为“凸出式”，“标题”属性设置为“九九表”。如图 7-10 所示。

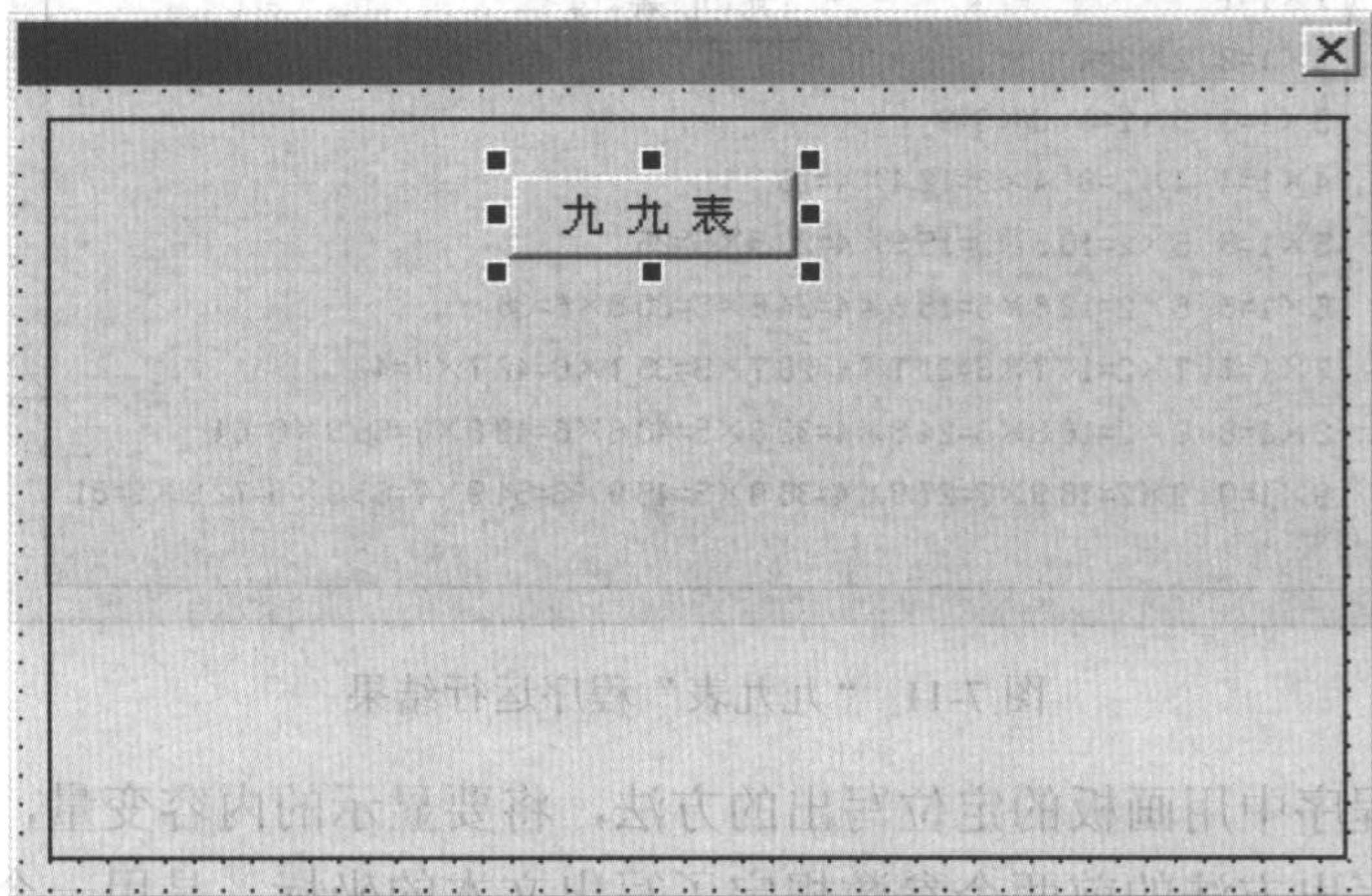


图 7-10 “九九表”程序界面

要想在画板上将九九表显示出来，首先考虑九九表中有哪些数，他们有什么规律，只要把九九表中每一个算式的乘数和被乘数依次取出，就可以得到要显示在画板上的内容，然后用画板组件的方法显示出来就可以了。在“\_标签 1\_被双击”事件子程序中添加代码。如下：

子程序名	返回值类型	公开	备注		
_标签1_被双击	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				





变量名	类型	静态	数组	备注
被乘数	整数型			
乘数	整数型			
要显示的内容	文本型			

--> 计次循环首 (9, 被乘数)

--> 变量循环首 (1, 被乘数, 1, 乘数)

要显示的内容 = 到文本 (被乘数) + "×" + 到文本 (乘数) + "=" + 到文本  
(被乘数 × 乘数)

画板1.定位写出 (乘数 × 45 - 40, 被乘数 × 20 - 10, 要显示的内容)

-- 变量循环尾 0

-- 计次循环尾 0

运行程序, 双击“标签”组件, 效果如图 7-11 所示。

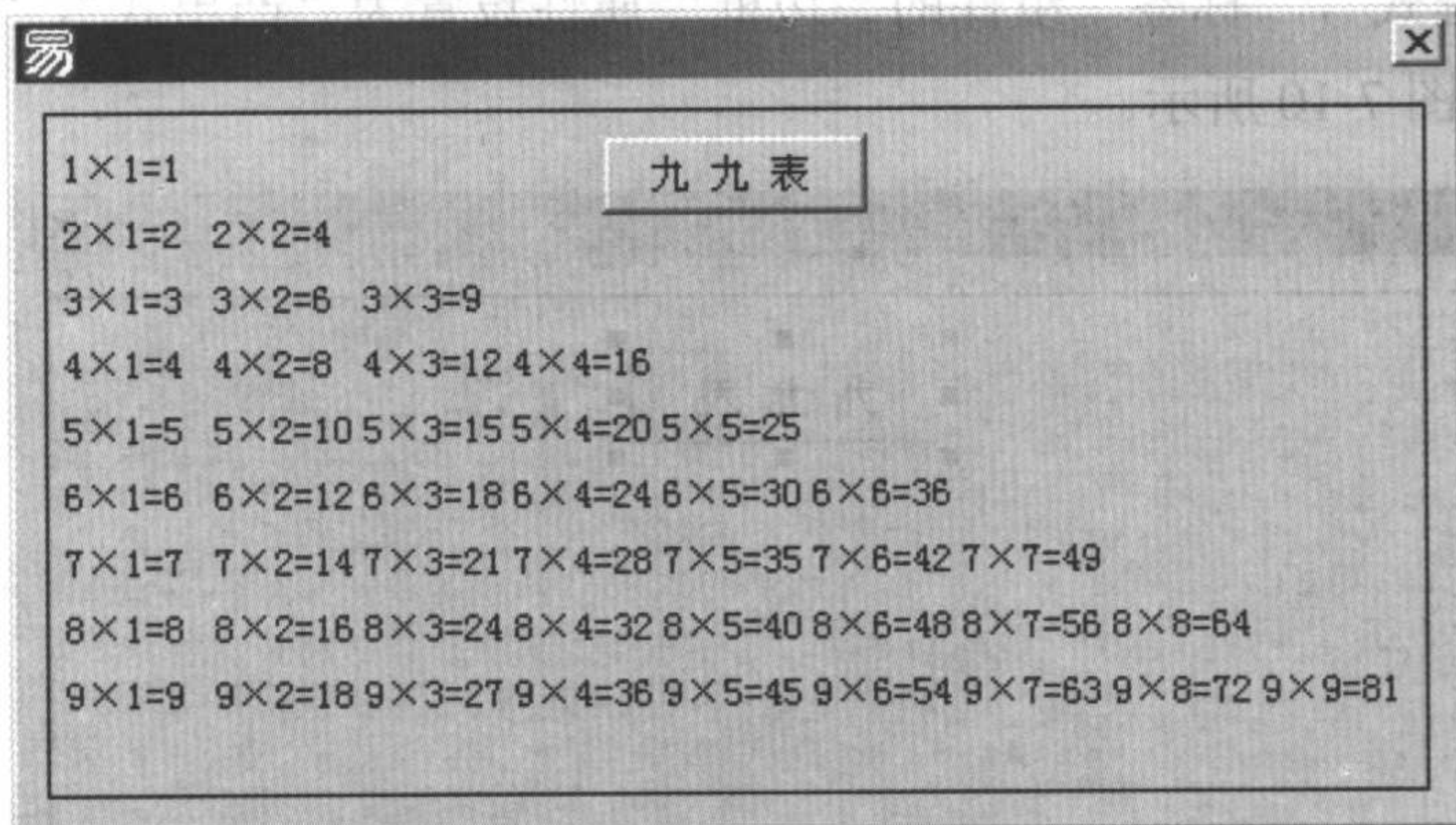


图 7-11 “九九表”程序运行结果

“九九表”程序中用画板的定位写出的方法, 将要显示的内容变量, 显示在画板的指定位置上, 定位写出方法的前两个参数规定了写出文本的坐标, 是用一个公式表示的, 只要了解定位写出方法的使用就可以了, 公式仅做了解。

“画板”的“画图片 ()”方法

图片号 = 载入图片 (#图片 1)

画板 1. 画图片 (图片号, 0, 0, , , )

画板 1. 底图 = #图片 2

以上两组代码的作用相同, 都是将一个资源表中的图片载入到画板。

下面两组代码可以将图片文件载入到画板。

如果真 (通用对话框 3. 打开 ())

图片号 = 载入图片 (通用对话框 3. 文件名)

画板 1. 画图片 (图片号, 10, 10, , , )

如果真结束

或

如果真 (通用对话框 3. 打开 ())

画板 1. 底图 = 读入文件 (通用对话框 3. 文件名)



如果真结束，更是要需对代码进行更全面的审查，并测试输出结果“画板”。不难看出，方法和命令在编程中的重要作用。

## 7.4 事件的触发

每个组件都有它的事件，组件的事件就是当一个组件状态发生了变化后，会触发相应的事件消息，并自动运行这个事件消息相应的子程序。比如，启动窗口的鼠标右键被按下事件，我们在相应事件子程序中输入了代码，程序运行以后，在启动窗口上点击鼠标右键，就会激活“\_启动窗口\_鼠标右键被按下”的子程序，并执行在这个子程序中输入的全部代码。

### 7.4.1 事件子程序

组件事件是由易语言开发环境已定义好的，可通过点击属性面板下方的事件下拉列表创建。创建位置如图 7-12 所示。

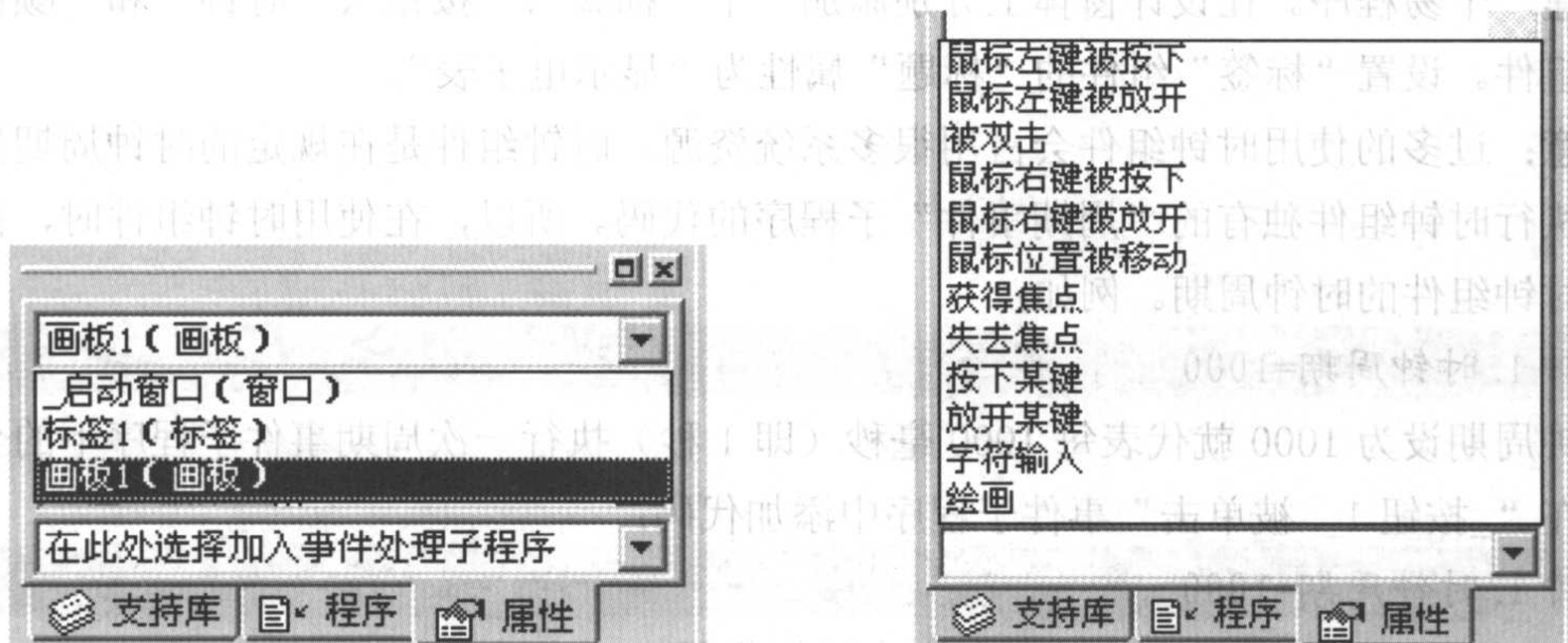


图 7-12 选择组件事件

新建一个易程序。添加一个“画板”组件。在属性面板事件栏中选择“绘画”，添加代码如下：

子程序名	返回值类型	公开	备注		
_画板1_绘画					
参数名	类型	参考	可空	数组	备注
重画区左边	整数型				
重画区上边	整数型				
重画区右边	整数型				
重画区下边	整数型				

- ▶ 画板1. 定位写出 (10, 10, “中文编辑工具-易语言!”)
- ▶ 画板1. 定位写出 (10, 30, “此处演示画板的‘绘画’事件。”)
- ▶ 画板1. 画渐变矩形 (0, 50, 360, 70, #从左到右, #红色, #黄色, #蓝色, #绿色)





“绘画”事件的触发时机：当画板中的全部或一部分区域需要被重新绘制时产生此事件。

该事件的处理子程序有四个参数，分别提供目前画板中待更新矩形区域的左上角、右下角坐标。使用中通常可忽略这些参数（即不设置它们）。

如果设置画板的“自动重画”属性为“假”（默认值），则所有对画板的绘图操作，都在本事件中进行。如当画板被其他窗口遮住后又显示时，程序自动产生“绘画”事件，自动调用“绘画”事件的“事件处理子程序”，重新绘制画板中的内容，以确保画板中的内容永远不会消失。

画板 1 的“自动重画”属性为“真”时，因为对画板的绘图操作是在“绘图”事件中进行的，所以即使画板被其他窗口遮住，画板中的内容也不会消失。

### 7.4.2 组件事件的应用

1. 本程序将每一秒显示当前时间到一个标签组件中，并可调整标签的背景颜色。

新建一个易程序。在设计窗体上分别添加一个“标签”、“按钮”、“时钟”和“颜色选择器”组件。设置“标签”组件的“标题”属性为“显示电子表”。

**注意：**过多的使用时钟组件会占用很多系统资源。时钟组件是在规定的时钟周期里，不断地执行时钟组件独有的“周期事件”子程序的代码。所以，在使用时钟组件时，都要先规定时钟组件的时钟周期。例如：

时钟 1. 时钟周期=1000

时钟周期设为 1000 就代表每 1000 毫秒（即 1 秒）执行一次周期事件子程序下的全部代码。在“\_按钮 1\_ 被单击”事件子程序中添加代码：

时钟 1. 时钟周期=1000

在“\_时钟 1\_ 周期事件”事件子程序中添加代码：

标签 1. 标题=到文本（取现行时间（））

按下按钮 1 后，每 1 秒钟时钟周期事件就会取一次现行时间，并用“到文本（）”命令转换“取现行时间（）”命令所返回的日期时间型数据，显示到“标签 1”的标题中。

为了让标签的颜色随着颜色选择器的改变而更换，要在“\_颜色选择器 1\_ 颜色被改变”事件触发时去调整标签的背景颜色。事件子程序的代码为：

标签 1. 背景颜色 = 颜色选择器 1. 颜色

将标签的背景颜色设置为颜色选择器的颜色。

按下 F5 键运行程序，并试着在颜色选择器中选择不同的颜色，查看运行效果。

2. 本程序将显示一个登录窗口，输入正确的用户名和口令后才可进入主界面。

新建易程序，在“\_启动窗口”中分别添加两个“标签”，两个“编辑框”和两个“按钮”组件。如图 7-13 所示。



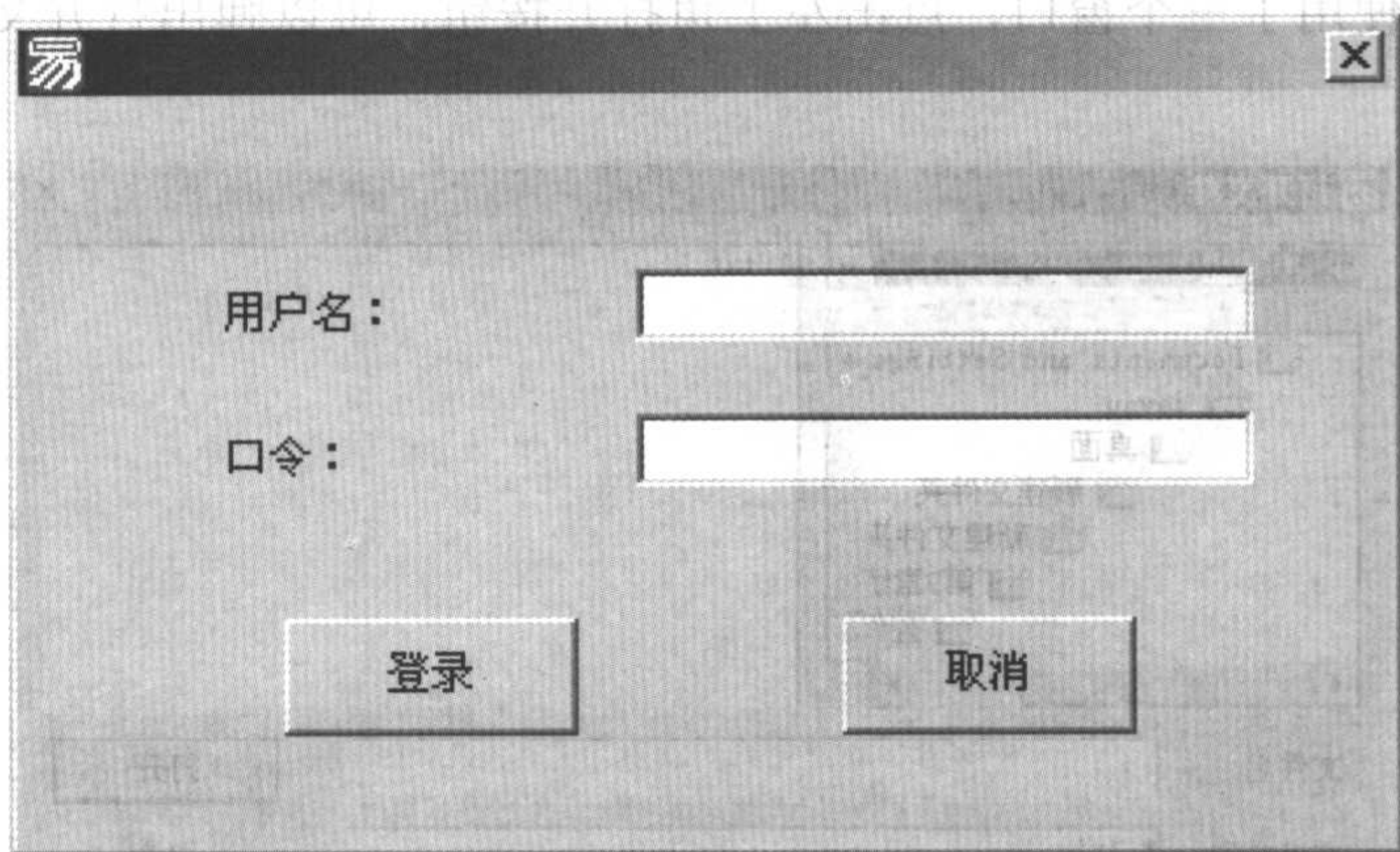


图 7-13 启动窗口设计界面

通过程序面板，插入“窗口 1”并在窗体上添加一个“标签”组件，将“标签”组件的“标题”属性设置为“程序的子窗口”，“字体”属性的“字体大小”改为“初号”。

回到启动窗口设计界面，在“\_按钮 1\_被单击”事件子程序中添加代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```

如果真 (编辑框1.内容 = “123” 且 编辑框2.内容 = “456”)
    _启动窗口.可视 = 假
    载入 (窗口1, , 真)
    
```

判断编辑框 1 的内容是否等于“123”并且编辑框 2 的内容是否等于“456”，如果两个条件都满足，就隐藏当前窗口，并载入窗口 1。（即判断用户名和口令是否正确。）

3. 一些组件需要配合使用，如：“驱动器框”、“目录框”、“文件框”组件。

打开书中例程：“打开文件与保存文件.e”。运行后主界面如图 7-14 所示。

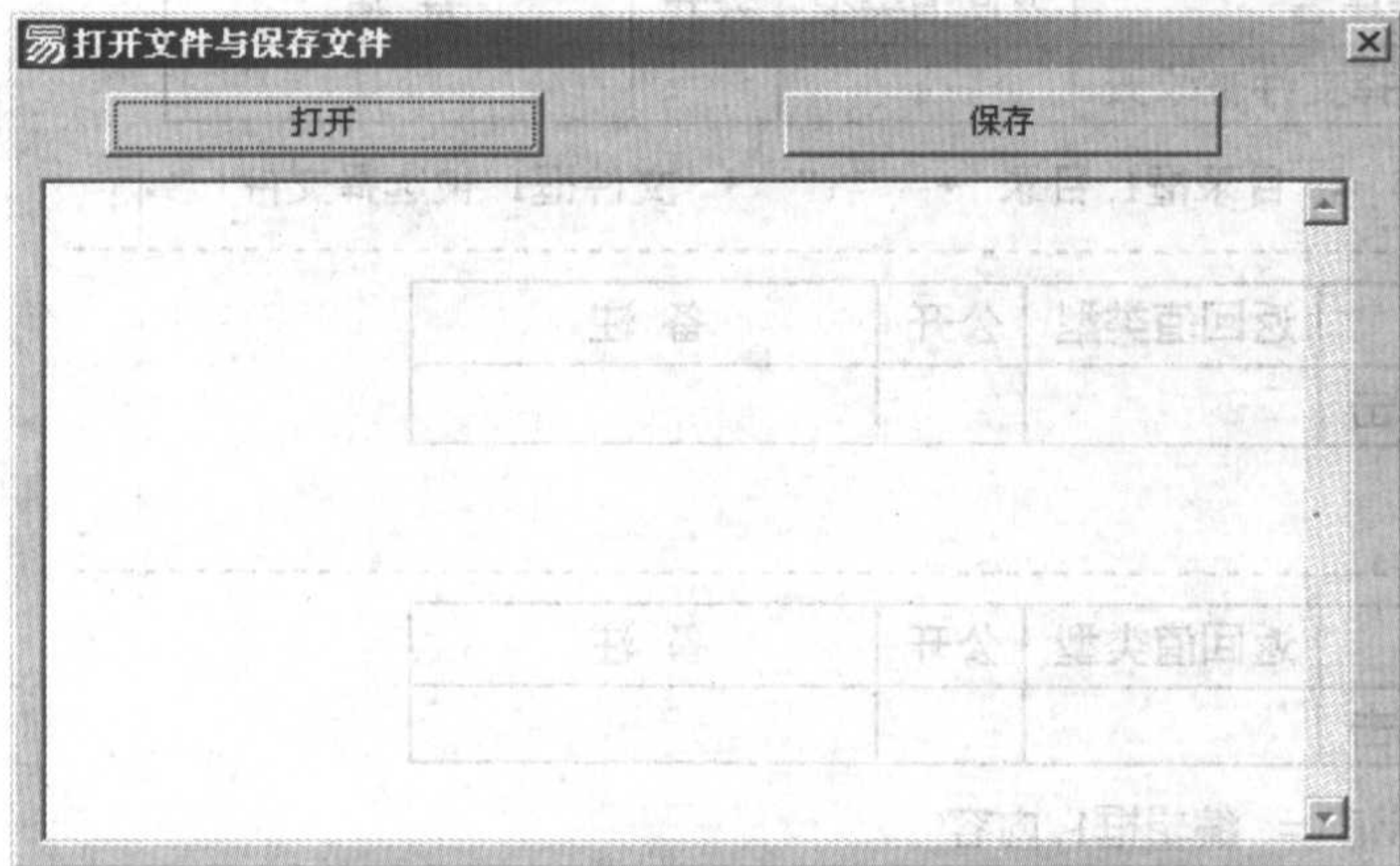


图 7-14 主界面共有三个组件





这个例程共使用了三个窗口，点击左上角打开按钮，可以弹出打开文件的新窗口。如图 7-15 所示。

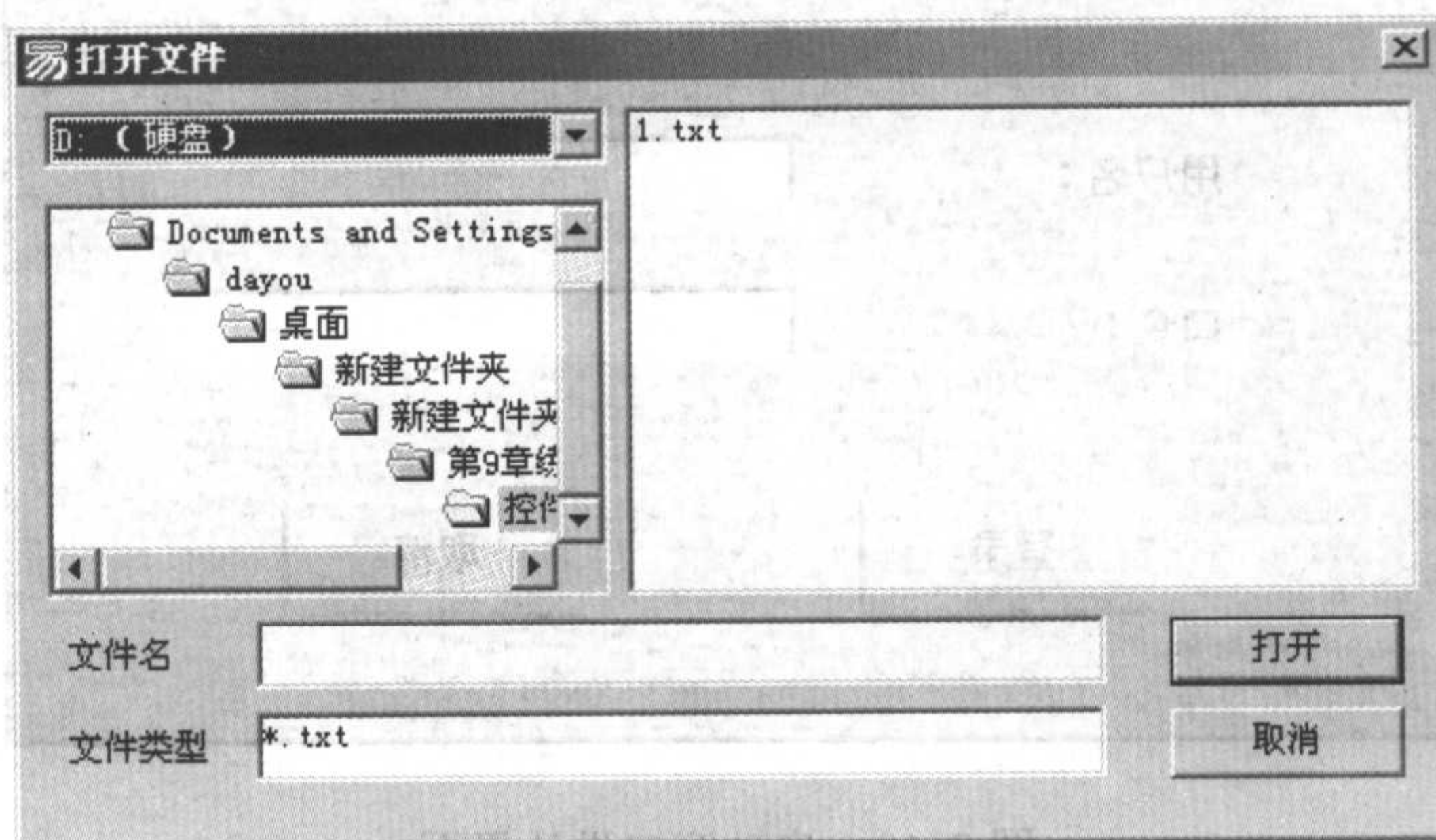


图 7-15 打开文件窗口

在打开文件窗口中，放置了“驱动器框”、“目录框”、“文件框”三个组件。相关程序代码如下：

窗口程序集名	备注
打开窗口程序集	

子程序名	返回值类型	公开	备注
_驱动器框1_驱动器被改变			

连续赋值 (驱动器框1.驱动器 + “:\”, 目录框1.目录, 文件框1.目录)

子程序名	返回值类型	公开	备注
_目录框1_目录被改变			

文件框1.目录 = 目录框1.目录

子程序名	返回值类型	公开	备注
_文件框1_选择文件被改变			

编辑框1.内容 = 目录框1.目录 + “\” + 文件框1.被选择文件

子程序名	返回值类型	公开	备注
_按钮2_被单击			

销毁 ()

子程序名	返回值类型	公开	备注
_按钮1_被单击			

\_启动窗口.标记 = 编辑框1.内容

销毁 ()



子程序名	返回值类型	公开	备注
_文件框1_双击选择			

编辑框1.内容 = 目录框1.目录 + “\” + 文件框1.被选择文件  
\_按钮1\_被单击 ()

其中,“\_文件框1\_双击选择”事件子程序可以在双击文件框组件中的文件名时,直接打开所选择的文件。

回到主界面后,点击右上角的保存按钮,可以弹出保存文件窗口。如图 7-16 所示。

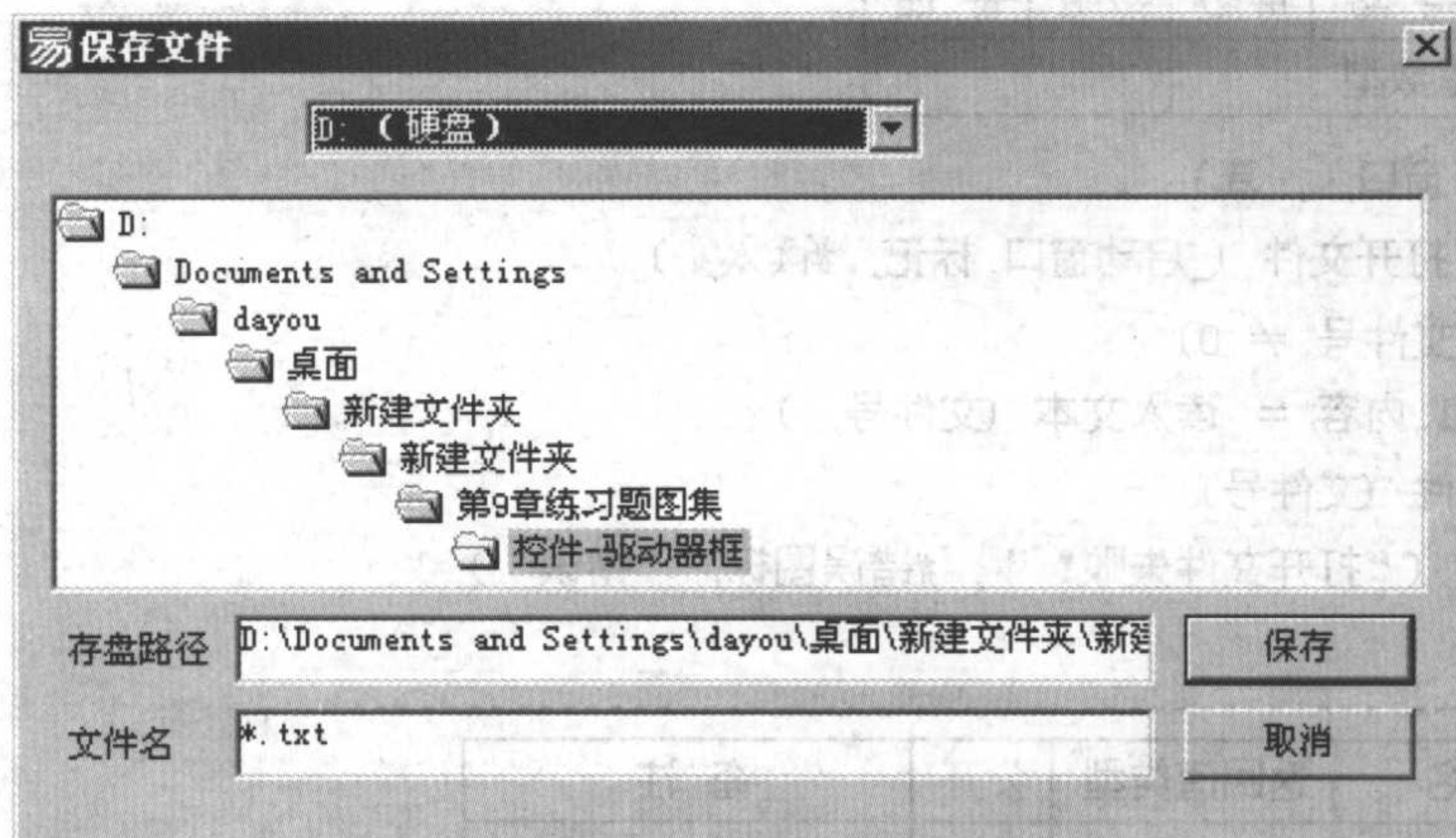


图 7-16 保存文件窗口

相关程序代码如下:

子程序名	返回值类型	公开	备注
_保存窗口_创建完毕			

编辑框1.内容 = 目录框1.目录

子程序名	返回值类型	公开	备注
_按钮2_被单击			

销毁 ()

子程序名	返回值类型	公开	备注
_按钮1_被单击			

\_启动窗口.标记 = 编辑框1.内容 + “\” + 编辑框2.内容  
销毁 ()

子程序名	返回值类型	公开	备注
_目录框1_目录被改变			

编辑框1.内容 = 目录框1.目录





在此，借用“\_启动窗口”的“标记”属性作为一个全局变量使用。

主窗口的程序代码如下：

窗口程序集名	备注
窗口程序集1	

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
文件号	整数型			

载入 (打开窗口, , 真)

文件号 = 打开文件 (启动窗口. 标记, #读入, )

--- 如果 (文件号 ≠ 0)

编辑框1. 内容 = 读入文本 (文件号, )

--- 关闭文件 (文件号)

▶ 信息框 (“打开文件失败!”, #错误图标, “错误”)

子程序名	返回值类型	公开	备注
_按钮2_被单击			

变量名	类型	静态	数组	备注
文件号	整数型			

载入 (保存窗口, , 真)

文件号 = 打开文件 (启动窗口. 标记, #重写, )

--- 如果 (文件号 ≠ 0)

--- 如果 (写出文本 (文件号, 编辑框1. 内容) = 真)

--- 关闭文件 (文件号)

▶ 关闭文件 (文件号)

▶ 信息框 (“保存文件失败!”, #错误图标, “错误”)

这样就完成了自己编写的打开与保存文件的程序。本例程只是告诉大家这三个组件是如何互相配合使用的，如果使用“浏览文件夹( )”这个命令，可以代替上述复杂的代码编写。



## 7.5 本章小结

本章的重点任务是让大家自己了解新的组件，通过常用组件的介绍，当需要了解新组件时，就可以先从组件的属性开始了解，实际上有很多属性是十分直观的，一看名字就可以了解，同时在提示面板中有相关属性的提示。

了解事件可以在属性面板下方的事件列表中查看，组件也有自己默认的事件，即大家双击一个组件，就会产生此事件。如：窗口设计时，某个按钮被双击后就会产生“被单击”事件。默认事件是最常用的组件事件。

了解组件的方法可以通过支持库面板中的“数据类型”树型表查看。当鼠标选中一个方法后，会在提示面板中有相关方法的提示。方法的使用基本上与命令类似。

大家还可以进行以下练习：

1. 组件共有属性是组件的基本属性，是每个可视组件都必需具有的，包括\_\_\_\_\_。

2. 说明“内容”属性与“标题”属性的不同点。

3. 学习向窗口中添加组件，并学习此组件的相关属性、方法、事件。



## 第八章 多媒体

随着软硬件技术的发展, 个人电脑不仅能做文字处理和数值运算, 而且还能够处理图像、音视频等多种媒体信息数据, 多媒体信息包括数值、文本、图形、图像、音乐、语音、动画和视频信息等多方面内容, 一般所说的多媒体处理主要是指图像、动画和音视频处理。

本章介绍易语言中音频播放命令、图形图像组件及视频播放组件。掌握本章所介绍的基本多媒体组件, 将对今后的程序开发、界面美化、功能增强等方面都有着很重要的意义。

在 Windows 系统中, 常用的图形、声音、动画、视频等文件的类型有很多, 下面列出了一些常见的音频、动画、图形图像及视频的文件类型。

### 1. 声音文件

**\*. WAV** 声音文件, Windows 中常用的格式, 安装声卡和 WAV 驱动程序后可由媒体播放器播放。

**\*. MID** 音序文件, 安装声卡及 MIDI 驱动程序后, 用媒体播放器播放。

**\*. Mp3** 可用媒体播放器播放。

### 2. 图形文件

**\*. BMP** Windows 中常用的图形格式。

**\*. JPG** 压缩的高质量图片。

**\*. GIF** 动画图形, 常用于网上传播的图形格式。

### 3. 动画文件

**\*. SWF** 动画图形, 常用于网上传播的图形格式, 可用 Flash 播放。

**\*. AVI** 视频文件由微软公司制订的标准, 通过 Video for Windows 或其更新的媒体播放器来播放。

### 4. 视频文件

**\*. Wmv** 是由微软所制定的网络串流多媒体标准格式。

**\*. mpeg** 压缩视频的基本格式。压缩方法是将视频信号分段取样, 压缩比很大。

**\*. rm** 由 RealNetworks 公司所制定的音频/视频压缩规范。



## 8.1 声音（音频）

### 8.1.1 媒体播放命令

#### 1. “播放音乐（）”

可以播放 .WAV、.MID 声音文件或相应格式的字节集声音数据、声音资源。

实例代码如下：

```
逻辑变量 = 播放音乐 ( "C:\Windows\音乐文件.WAV" , 真 )
```

或

```
逻辑变量 = 播放音乐 ( #声音文件 , 真 )
```

其中“#声音文件”为声音资源。

第一个参数值为 .WAV、.MID 声音文件名称或相应格式的字节集声音数据、声音资源。

第二个参数值为“真”表示指定音乐将被循环播放，否则仅只播放一次。如果本参数被省略，默认为仅播放一次。

#### 2. “播放 MID（）”命令

可以自动连续播放多个 MIDI 声音文件（注意不支持 WAV）或相应格式的字节集声音数据、声音资源。

实例代码如下：

```
播放 MID ( , , 读入文件 ( "C:\Windows\音乐文件.MID" ) , #声音文件 )
```

或

```
加入成员 ( 字节集数组变量 , 读文件 ( "C:\Windows\音乐文件.MID" ) )
```

```
加入成员 ( 字节集数组变量 , #声音文件 )
```

```
播放 MID ( , , 读入文件 ( "C:\Windows\音乐文件.MID" ) , 字节集数组变量 )
```

其中，第 1 个参数为“播放次数”，第 2 个参数为“间隔时间”，第 3 个参数为“欲播放的 MIDI 音乐”。被播放的音乐文件可以连续扩充，并按顺序播放。这里的音乐文件必须转换为字节集型数据，也可以提供保存了多个字节集型音乐文件的字节集型数组。

#### 3. “播放 MP3（）”命令

可以自动连续播放多个 MP3 音乐文件。

实例代码如下：

```
播放 MP3 ( 1, "C:\Windows\音乐文件 1.MP3" , "C:\Windows\音乐文件 2.MP3" )
```

或

```
加入成员 ( 文本数组变量 , "C:\Windows\音乐文件 1.MP3" )
```

```
加入成员 ( 文本数组变量 , "C:\Windows\音乐文件 2.MP3" )
```

```
播放 MP3 ( , "C:\Windows\音乐文件.MP3" , 文本数组变量 )
```





其中，第 1 个参数为“播放次数”，第 2 个参数为“欲播放的 MP3 文件名”。文本数组变量保存的是多个 MP3 音乐文件的文件名称。注意和“播放 MID ()”命令的区别。

下面举例说明。打开“mp3 播放者.e”例程。界面如图 8-1 所示。

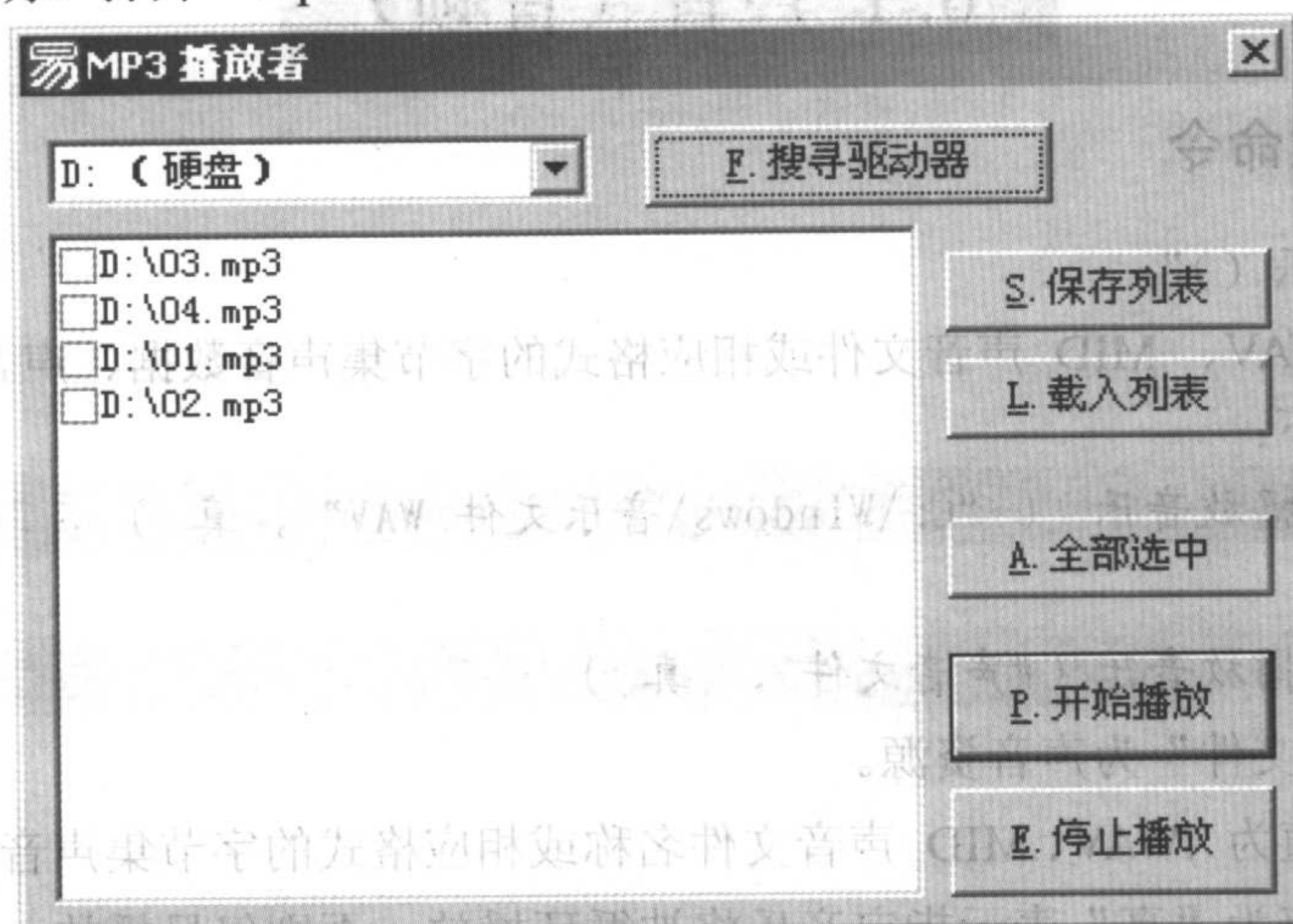


图 8-1 程序界面

点击名称为“搜寻驱动器”的按钮，将自动搜索被选择驱动器中“MP3”文件，并将文件路径加入到选择列表框中。在选择列表框中逐一选择或点击“全部选中”按钮选择欲播放的 MP3 文件，点击“开始播放”按钮将根据所选择的播放列表播放音乐。

“\_开始播放按钮\_被单击”事件子程序中的代码如下：

变量名	类型	静态	数组	备注
待播放MP3	文本型		0	
计次变量	整数型			

```
--> 计次循环首 (选择列表框1.取项目数 0, 计次变量)
--> 如果真 (选择列表框1.是否被选中 (计次变量 - 1) = 真)
--> 加入成员 (待播放MP3, 选择列表框1.取项目文本 (计次变量 - 1))
--> 计次循环尾 0
--> 如果真 (取数组成员数 (待播放MP3) > 0)
--> 播放MP3 (-1, 待播放MP3)
```

这段代码的作用是：使用计次循环命令将选择列表框中被选中项目的项目文本（MP3 音乐文件绝对路径）加入到文本数组“待播放 MP3”中，建立播放列表。然后判断歌曲列表中歌曲数量是否大于 0，如果是，就用“播放 MP3 ()”命令循环播放加入列表的 MP3 文件。



## 8.2 图片处理（图形图像）

### 8.2.1 图片的合并

画板组件功能十分强大，通过与其他方法和命令配合使用，可以用“画板”组件对图片进行多种处理（如：合并图片、切割图片等）。下面举例说明合并图片的方法。

打开例程“图片组处理.e”。程序运行界面如图 8-2 所示。

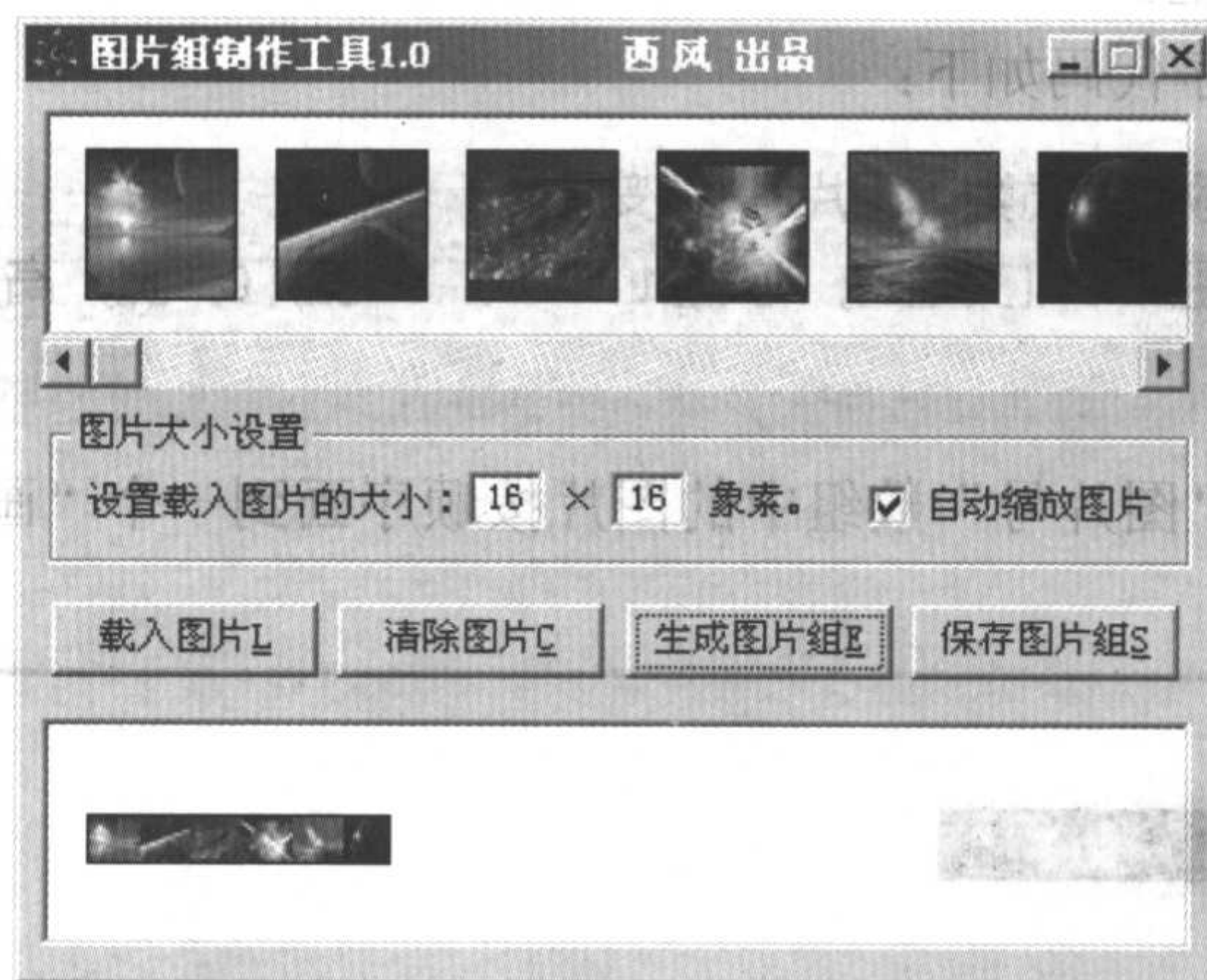


图 8-2 例程运行界面

这个程序的主要功能都是由画板实现的。关键部分代码如下：

```

--- 如果真 (通用对话框1. 打开 ())
    当前图片数 = 取数组成员数 (图片画板)
    重定义数组 (图片号, 真, 当前图片数 + 1)
    图片号 [当前图片数 + 1] = 载入图片 (通用对话框1. 文件名)

```

当“通用对话框”被打开，用户可选择图片文件，因为增加了一张新的图片，所以图片数量有所改变，需要将变量“当前图片数”加 1，第 2 行使用“取数组成员数 ()”命令取得原有多少张图片，在第 3 行程序中将“图片号”增加 1。程序调用“载入图片 ()”命令为此图片分配资源，并把此图片文件的图片号保存到“图片号”数组中。

```

复制窗口组件 (画板2, 图片画板 [当前图片数 + 1])
图片画板 [当前图片数 + 1]. 移动 (当前图片数 × 48 + (当前图片数 + 1) × 12, 10, 48, 48)
图片画板 [当前图片数 + 1]. 可视 = 真
图片画板 [当前图片数 + 1]. 画图片 (图片号 [当前图片数 + 1], 0, 0, 48, 48, 8, )

```

上述代码每载入一幅图片就动态创建一个“画板”组件，并将图片显示其上。第 1 行





为复制一个新的画板用于显示新图片，第 2 行为移动新画板到正确的位置，第 3 行为让此新画板可视，第 4 行为画出图片到这个画板中显示。如图 8-3 所示。

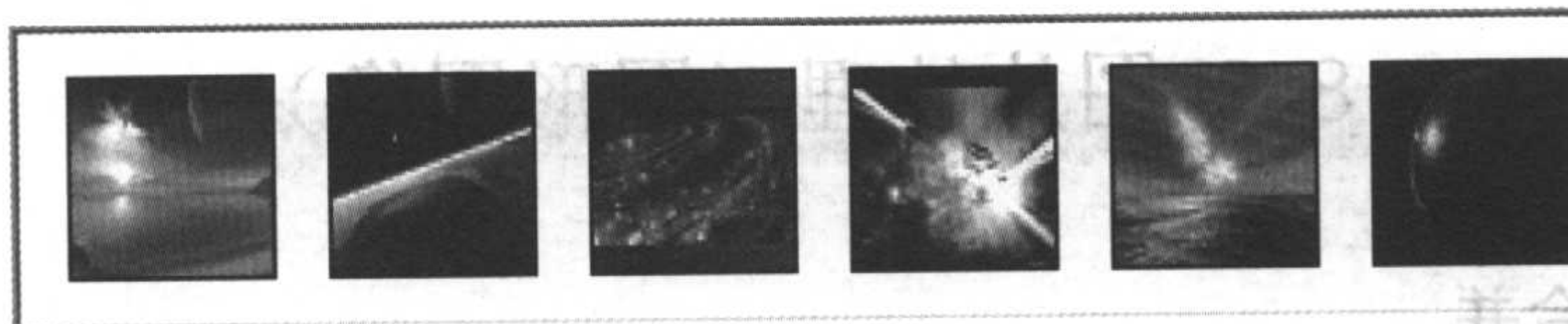


图 8-3 用复制出来的画板显示图片

注意：若只需要简单实现显示图片功能，完全可以直接把所有图片用“画板.画图片( )”画在一个“画板”组件上。

生成合并图片的程序代码如下：

```
-- 计次循环首 (取数组成员数 (图片号), 变量1)
    画板4.画图片 (图片号 [变量1], (变量1 - 1) × 宽, 0, 宽, 高, )
-- 计次循环尾 ()
```

上述代码把保存在“图片号”数组中的图片按顺序画到一个“画板”组件中。如图 8-4 所示。

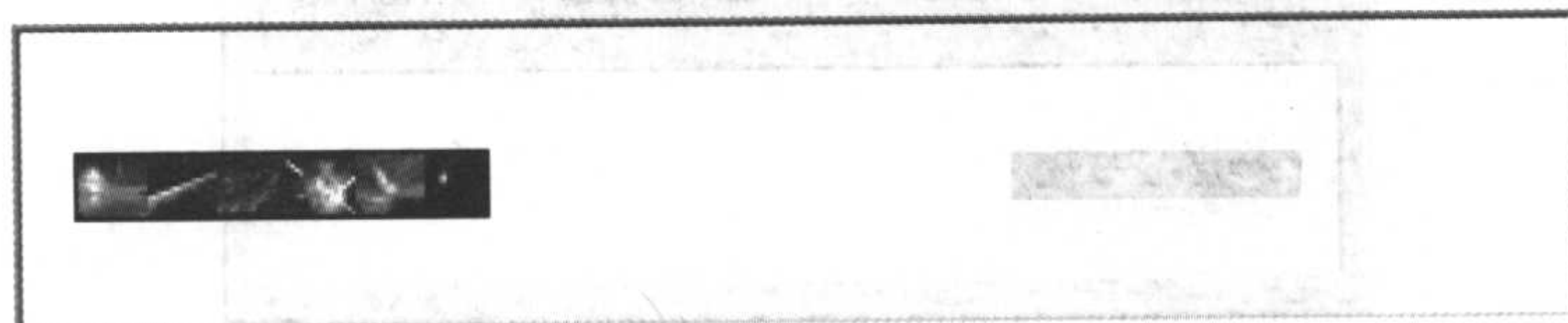


图 8-4 合并后的图片

用“写到文件( )”命令把“画板”组件中的图片保存起来。程序代码如下：

子程序名	返回值类型	公开	备注
_按钮3_被单击			

```
-- 如果真 (通用对话框2. 打开 ())
-- 如果 (写到文件 (通用对话框2. 文件名, 画板4. 取图片 ( )))
    信息框 (“保存图片组文件” + 通用对话框2. 文件名 + “成功!”, #信息
        图标, “保存图片组”)
    信息框 (“保存图片组文件” + 通用对话框2. 文件名 + “失败!”, #错误
        图标, “保存图片组”)
-- 结束
```

上述代码实现了保存图片功能。

## 8.2.2 图片框组件

图片框的重要属性有：“图片”、“显示方式”、“播放动画”。

### 1. 图片框组件的属性

#### (1) “图片”属性



图片框组件的“图片”属性指定要显示的图片，支持 BMP、JPG、GIF、ICO、CUR 图片格式。可在界面设计时直接选择磁盘上的图片文件，导入至此属性中，或保存到图片资源中，作为图形资源动态引用。程序代码如下：

```
图片框 1. 图片 = #图片 1
```

也可在运行期间读取硬盘上的一个图片文件，程序代码如下：

```
图片框 1. 图片 = 读入文件(文件名)
```

### (2) “显示方式”属性

“显示方式”属性控制图片在图片框中的显示方式。类型为**整数型**，可取以下值之一：“0. 图片居左上”、“1. 缩放图片”、“2. 图片居中”。默认为“0. 图片居左上”。

在这里需要说明的是，它有一个特殊的显示方式，即“缩放图片”功能。当图片框的高度与宽度变化时，图片会自动适应这种变化，随之改变大小。

### (3) “播放动画”属性

“播放动画”属性可控制播放 GIF 动画。为**逻辑型**，只能为“真”或“假”，默认为“真”。在代码中设置该属性，可以控制动画的播放与否。本属性只在“图片”属性为 GIF 格式时有效。

## 2. 图片框组件实例

图片框既可以显示图片，也可以保存图片框中的图片文件。用“写出字节集( )”命令来保存图片框中的图片到文件。实例代码如下（“图片框 1”中已经载入了一幅 BMP 位图）：

```
通用对话框 1. 类型 = 1
```

```
通用对话框 1. 默认文件后缀 = “.bmp”
```

```
通用对话框 1. 过滤器 = “*.bmp|*.bmp”
```

```
写出字节集 (打开文件 (通用对话框 1. 文件名, #改写, ), 图片框 1. 图片)
```

上述代码中的第 1 行实现了设置通用对话框的类型为“保存”对话框，第 2 行设置此通用对话框的默认后缀为 BMP，第 3 行设置了通用对话框的过滤器，第 4 行将图片框中的内容保存到用户指定的文件中。

图片文件也可以保存到数据库中。打开例程：“人事档案管理系统.e”，可以看到这个例程右上角有一个图片框可以显示图片。使用鼠标右键点击后可以弹出一个菜单，点击此菜单选项，就可以将外部图片填充至这个图片框之中。程序界面如图 8-5 所示。

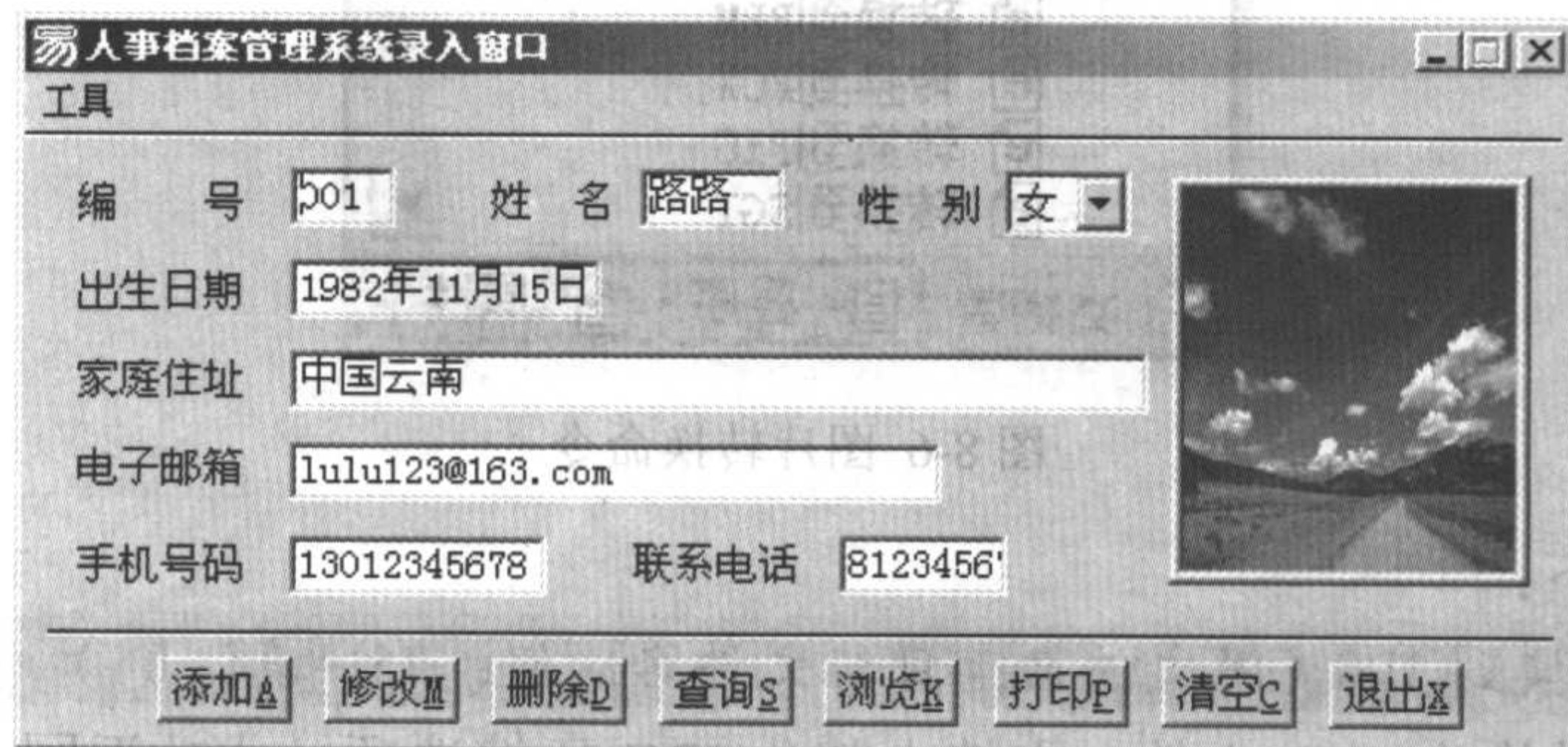


图 8-5 例程界面





图 (1) 显示图片

显示空白图片框或清空图片框的程序代码如下:

```
图片框.图片 = { }
```

显示易数据库中图片的程序代码如下:

```
图片框.图片 = 读 ( "图库 1" ) // "图库 1" 为数据库文件字段
```

(2) 保存图片到数据库中

在上述例程的修改按钮中, 要将图片框中的图片保存到易数据库中所用到的程序代码如下:

```
修改 ( , 图片框 1.图片 )
```

注意: 对应图片字段的字段类型必须设为字节集类型, 否则保存图片会失败。

3. 图片框组件与画板组件的功能和属性对比。如下表 8-1 所示。

	动画	缩放图片	文字	绘画	数据源
图片框	可以	可以	不能, 文字先要处理为 图片	无	有
画板	不可以	可以	可以	有	没有

表 8-1 图片框组件与画板组件的功能属性对比表

从表中可以看出, 图片框着重于显示实际存在的图片, 而且在动画与数据库的支持上比较好。而画板组件着重于图形的处理, 更像是一个绘画的工具。

#### 4. 图片转换

易语言中还提供了专门的图片转换命令。包括“转换到 JPG ()”、“转换到 BMP ()”、“转换到 TIF ()”、“转换到 PNG ()”、“转换到 PPM ()”、“转换到 PGM ()”、“转换到 PBM ()”、“转换到 PCX ()”、“转换到 PIC ()”、“转换到 SGI ()”命令。如图 8-6 所示。

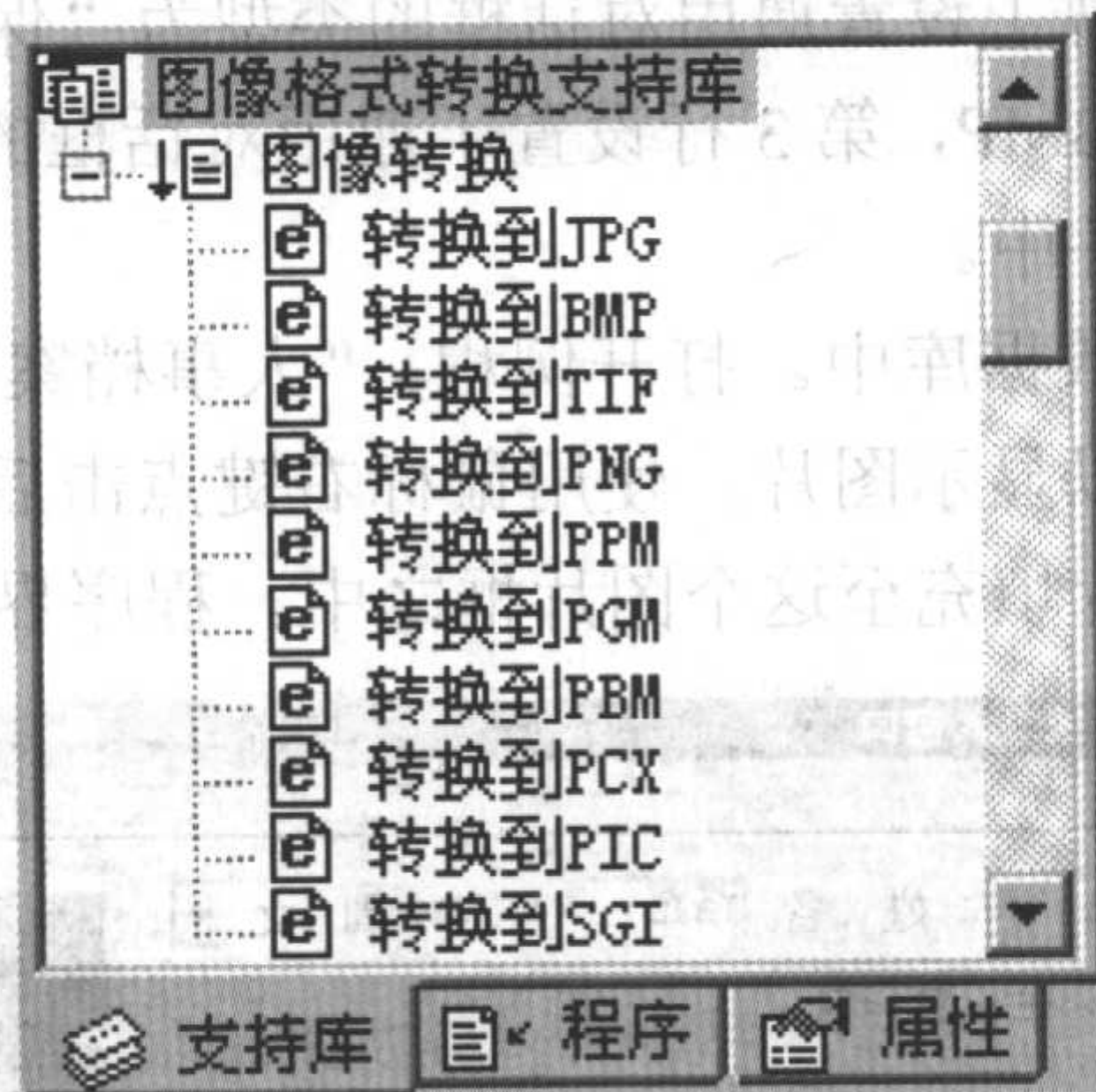


图 8-6 图片转换命令

实例代码如下:

```
整数变量=转换到 JPG (源文件名, 目标文件名, 假, 75, 真, 0 )
```

将图像文件转换为 JPG 文件, 支持大部分 JPG 压缩选项。成功返回 1, 失败返回一个小于等于零的数值, 具体为: 0: 未知错误; -1: 文件不存在; -2: 系统资源不足; -3:



该文件格式不被支持； -4: 编码时出错； -6: 编码时系统资源不足； -7: 编码参数错误。

在转换过程中要注意返回值，以便于错误判断处理。

例程“易语言图片格式转换.e”。利用以上命令转换图片。程序界面如图 8-7 所示。

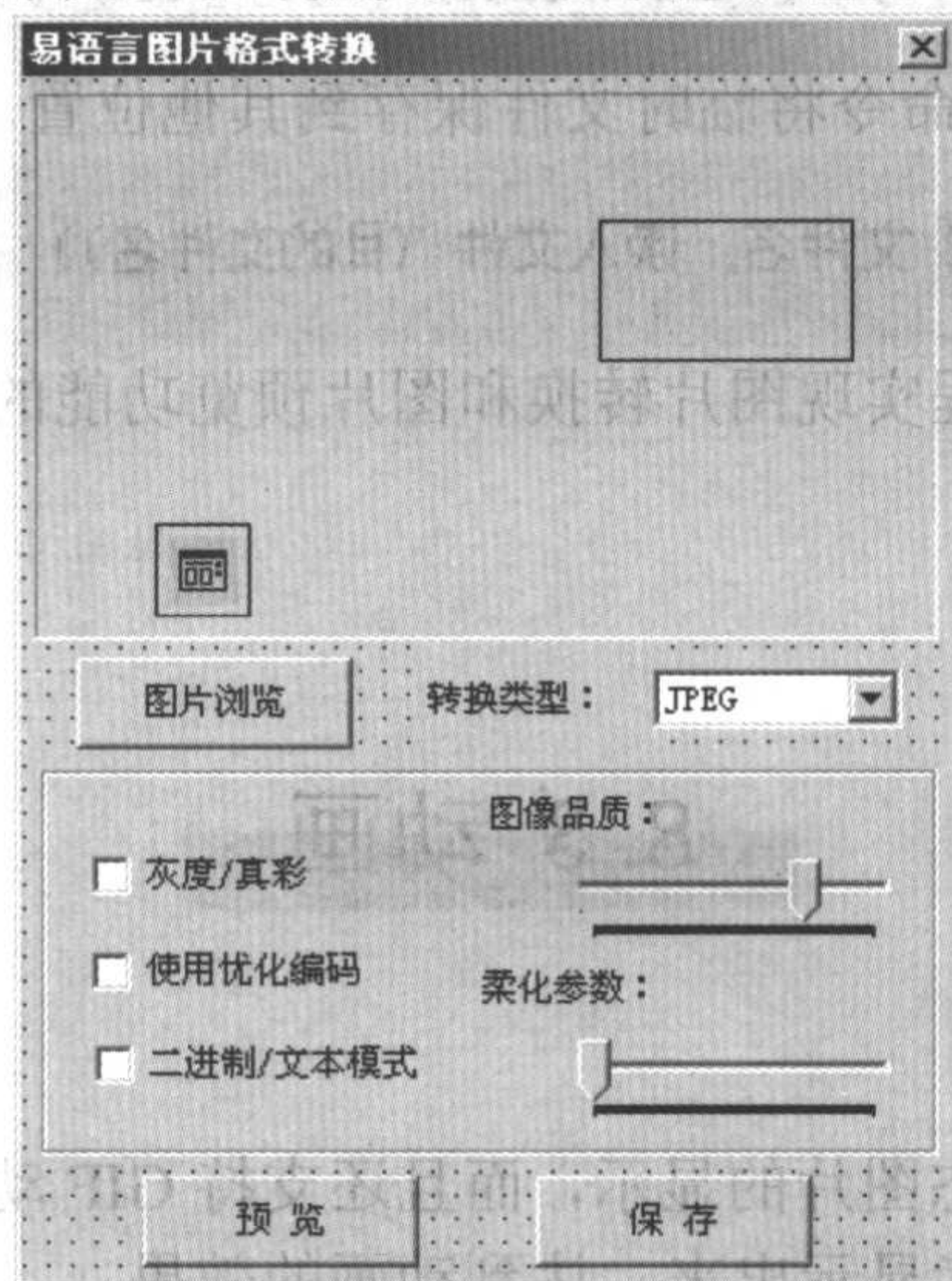


图 8-7 程序设计界面

此程序用两个“画板”组件分别显示转换前和转换后的图片。“\_按钮 1\_被单击”事件子程序中相关代码如下：

```
源文件名 = 通用对话框1.文件名
文件号 = 载入图片 (源文件名)
画板1.画图片 (文件号, 0, 0, 画板1.宽度, 画板1.高度, )
```

用“画板 1.画图片 ()”把原图片显示到“画板”组件中以便查看转换前效果。用图片转换命令转换原图片文件并保存为一个临时文件，“\_按钮 2\_被单击”事件子程序中相关程序代码如下：

```
-- 判断 (组合框1.现行选中项 = 0)
目的文件名 = 目的文件名 + 选中项
错误信息 = 转换到JPG (源文件名, 目的文件名, 选择框1.选中, 滑
块条1.位置, 选择框2.选中, 滑块条2.位置)
-- 判断 (组合框1.现行选中项 = 1)
目的文件名 = 目的文件名 + 选中项
-- 错误信息 = 转换到BMP (源文件名, 目的文件名)
```

“\_组合框 1\_列表项被选择”事件子程序代码如下：

```
选中项 = 组合框1.取项目文本 (组合框1.现行选中项)
```

用画板 2 显示被保存的临时文件以便查看转换后效果。“\_按钮 2\_被单击”事件子程序





相关代码如下：

```
文件号 = 载入图片 (目的文件名)  
画板2.画图片 (文件号, 0, 0, 画板2.宽度, 画板2.高度, )
```

最后用“写到文件 ( )”命令将临时文件保存到其他位置。程序代码如下：

```
写到文件 (通用对话框2.文件名, 读入文件 (目的文件名))
```

以上程序代码是这个例程实现图片转换和图片预览功能的主要片段，完整程序代码请参见实际例程。

## 8.3 动画

### 8.3.1 图片框的 GIF 动画

图片框不但支持多种静态图片的显示，而且还支持 GIF 动画。当 GIF 文件被浏览查看时，会自动按顺序把图片连续显示出来，达到动画的效果。

例程“图片框属性-图片&显示方式&播放动画.e”运行效果如图 8-8 所示。

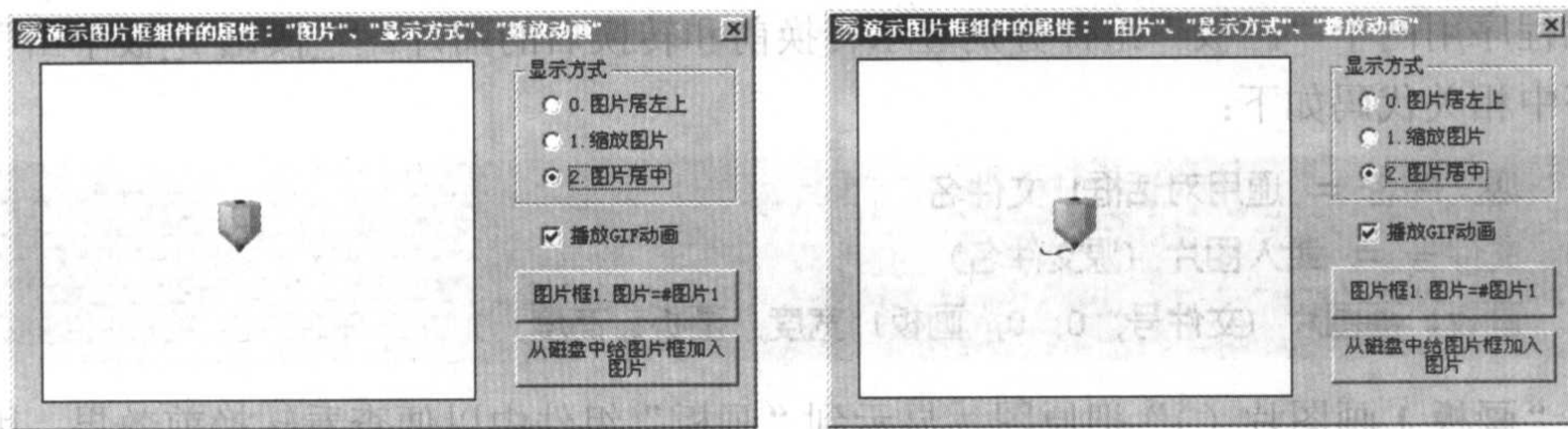


图 8-8 图片框显示 GIF 动画

### 8.3.2 窗口动画

#### 1. 帧动画窗口

例程“动画窗体.e”。程序运行效果如图 8-9 所示。

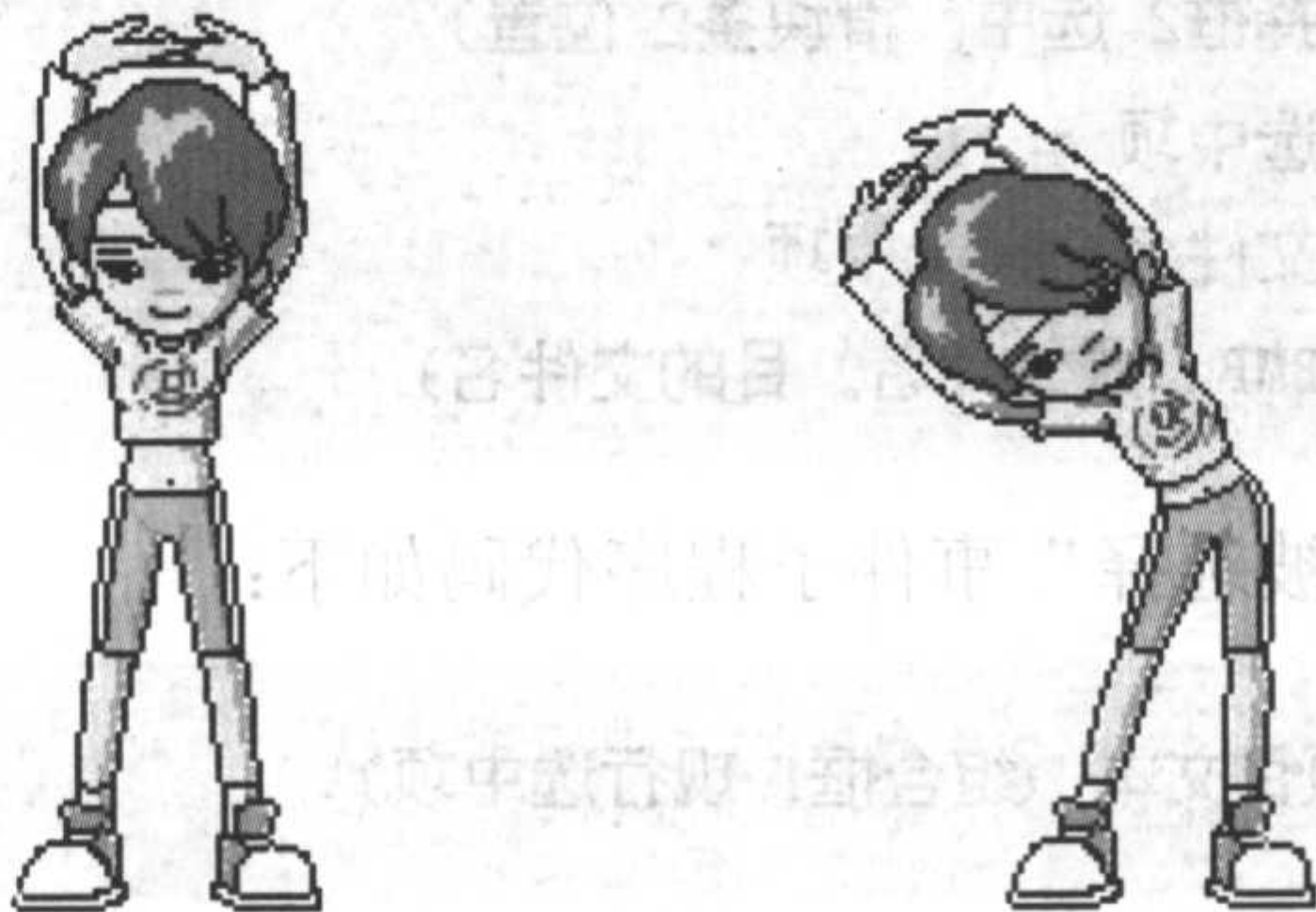


图 8-9 窗口置外形图片



“\_启动窗口\_创建完毕”事件子程序代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

变量名	类型	静态	数组	备注
循环变量	整数型			

```

--▶ 计次循环首 (13, 循环变量)
    GIF图片 [循环变量] = 多项选择 (循环变量, #图片1, #图片2, #图片3, #图片4, #图片5,
        #图片6, #图片7, #图片8, #图片9, #图片10, #图片11, #图片12, #图片13)
    帧延时 [循环变量] = 多项选择 (循环变量, 160, 150, 150, 600, 150, 150, 180, 150,
        150, 600, 150, 150, 800)
-- 计次循环尾 ()
帧数 = 1
_时钟1_周期事件 ()

```

以上代码的作用是把图片资源保存到字节集数组中，并把各图片的间隔时间保存到整数数组中，然后调用“\_时钟1\_周期事件 ()”事件子程序依次显示。“\_时钟1\_周期事件”事件子程序的代码如下：

```

-- 如果真 (帧数 > 13)
    帧数 = 1
    _启动窗口.置外形图片 (GIF图片 [帧数], #白色)
    _启动窗口.底图 = GIF图片 [帧数]
    时钟1.时钟周期 = 帧延时 [帧数]
    帧数 = 帧数 + 1

```

“如果真 ()”条件语句判断上一次的图片是否是图片组中的最后一张图片，如果已显示到最后一张图片，就把当前图片置为第一张图片，以循环播放。“帧数”保存的只是图片组的索引。然后用“\_启动窗口.置外形图片 ()”方法把不需要的背景色置为透明色，同时改变“时钟1”的时钟周期，依次读取下一张图片。

## 2. GIF 窗口

打开“动画窗体\_演示.e”。这个例程只调用了一个易模块。在运行程序之前请首先用易模块管理器导入名为“动画窗体 1.3.ec”的模块。程序调用代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

动画窗体 (\_启动窗口)

程序运行效果如图 8-10 所示。

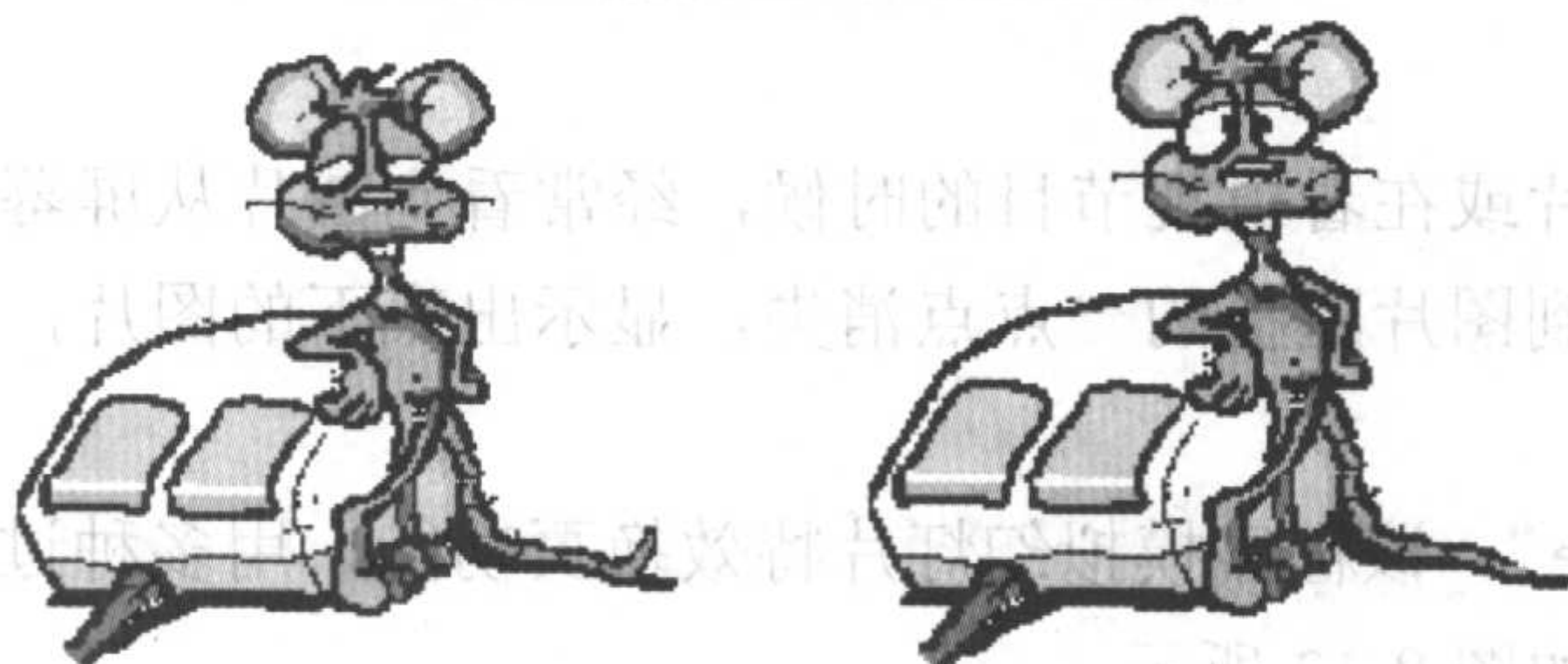


图 8-10 程序运行效果





通过模块的处理，GIF 动画图片就可以正常显示。

### 3. SWF 文件的播放

例程“SWF 播放.e”用超文本浏览框播放 SWF 文件。程序运行效果如图 8-11 所示。

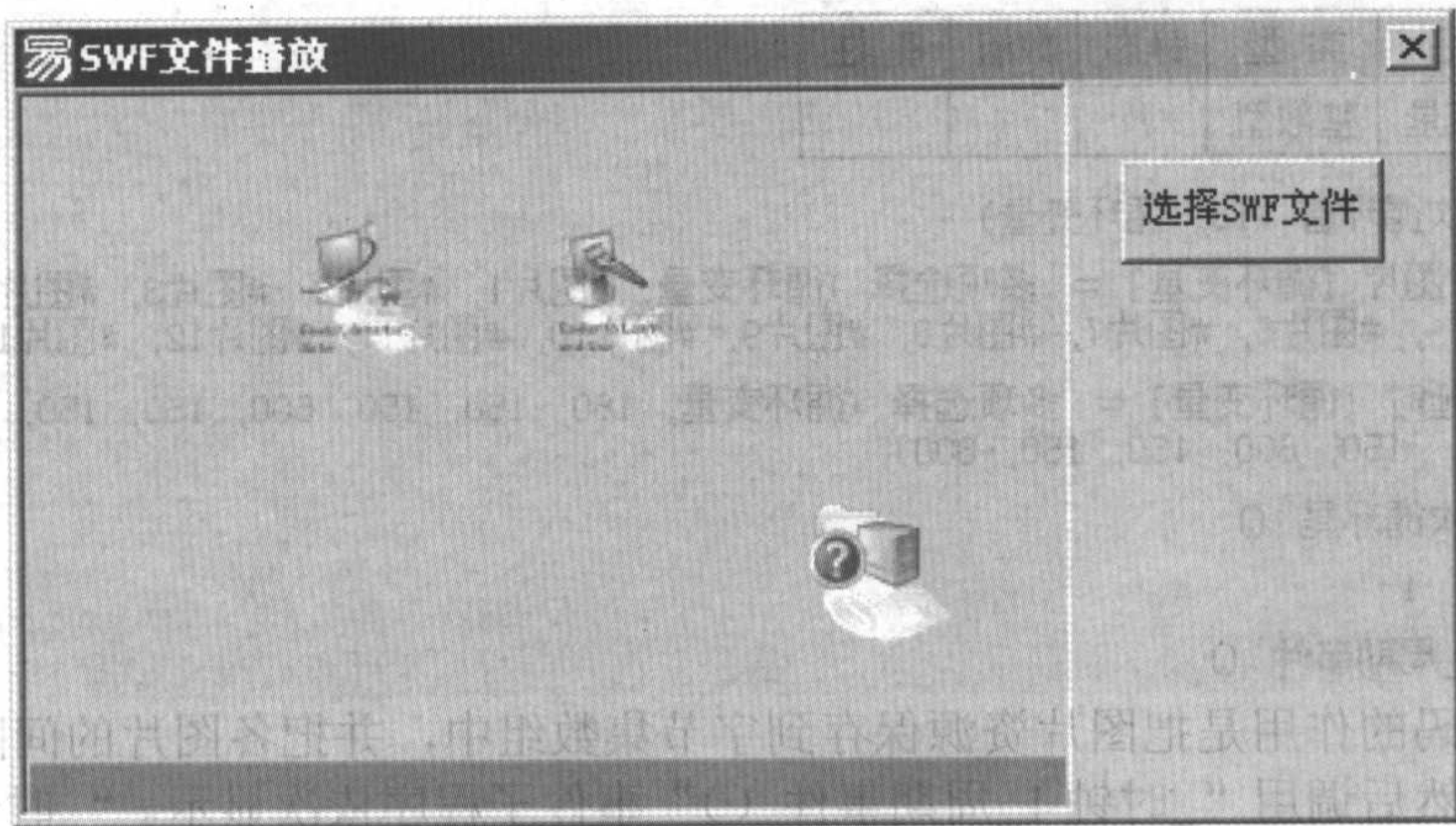


图 8-11 例程运行界面

通过“\_图形按钮 1\_被单击”事件子程序选择扩展名为 SWF 的文件，然后用“超文本浏览框”播放。程序代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```
通用对话框1.过滤器 = "SWF文件 (*.swf) | *.swf"  
如果 (通用对话框1.打开 () = 真)  
    超文本浏览框1.跳转 (通用对话框1.文件名, , )  
    超文本浏览框1.跳转 (取特定目录 (10) + "Help\Tours\mm  
        Tour\intro.swf", , )
```

程序用“超文本浏览框 1.跳转 ()”命令获取并播放 SWF 文件。

## 8.4 图片转场

大家在观看幻灯片或在看电视节目的时候，经常看到图片从屏幕外以多种方式慢慢移动到屏幕中间，或看到图片随机的一点点消失，显示出底下的图片，现在就用程序来模拟这种特效。

打开例程“转场.e”。该程序模拟幻灯片特效换页功能，用多种过渡方式实现图片转场显示。程序运行效果如图 8-12 所示。



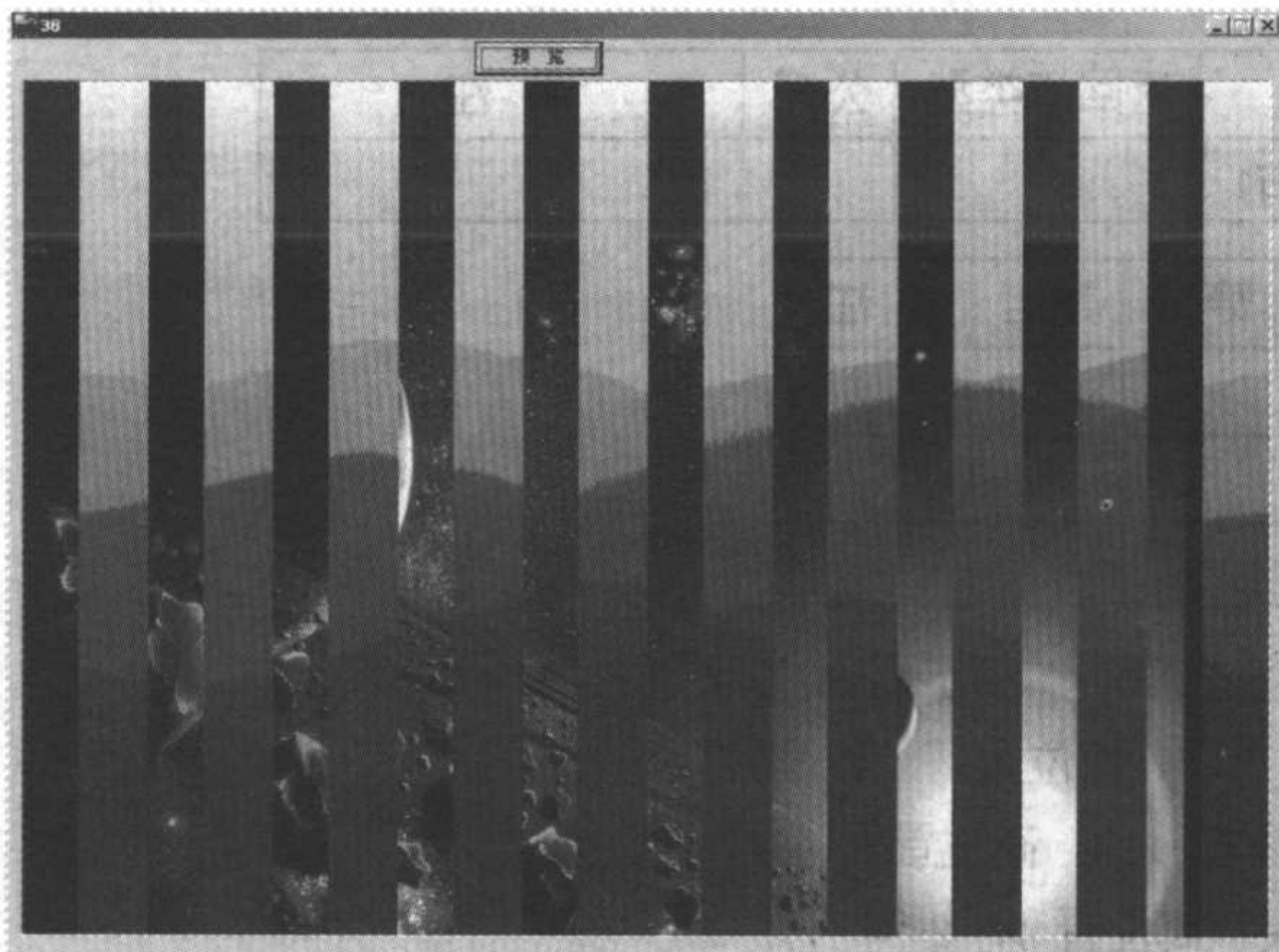


图 8-12 百叶窗式转场

特效程序运行的效果好坏，最关键的就是其特效算法的设计。由于例程一共包含十类 48 种特效，程序代码量比较大，因此仅就个别典型算法做讲解，其余特效实现算法请对照例程自行分析。

在这里，本书省略对窗口设计与初始化代码的讲解，直接来看特效算法。需要说明的是，每当一个特效演示完毕后，程序都会自动更换隐藏画板的背景图片，因此在算法代码部分，默认隐藏画板已经填充好第二张图片了。

“推出效果之从左往右 1”子程序的代码如下：

子程序名	返回值类型	公开	备注
推出效果之从左往右1			从左往右

变量名	类型	静态	数组	备注
左边1	整数型			

--> 判断循环首 (左边1 < 演示画板.宽度 + 5)

演示画板.复制 (左边1, 0, 演示画板.宽度 - 左边1, 演示画板.高度, , 左边1 + 5, 0, )

左边1 = 左边1 + 5

隐藏画板.复制 (隐藏画板.宽度 - 左边1, 0, 左边1, 隐藏画板.高度, 演示画板, 0, 0, )

延时 (速度)

-- 判断循环尾 ()

变量“左边 1”记录前景图片（即演示画板中的底图）向右移动的位置。每一次循环，演示画板都会将前景图片从当前位置向右偏移 5 个像素再画回原画板，并使当前位置也增加 5 个像素；隐藏画板则将自身图片画到演示画板空白处，直到当前位置等于画板最右边，形成推动效果。当然，这里所说的“空白处”并不是真正的清除了图片，借指演示画板的当前位置左边部分。

“棋盘效果之从左往右 1”子程序的代码如下：





子程序名	返回值类型	公开	备注
棋盘效果之从左往右1			

变量名	类型	静态	数组	备注
棋盘宽度	整数型			
宽位置	整数型			
高位置	整数型			

```
--> 判断循环首 (棋盘宽度 < 演示画板.宽度 ÷ 10 + 2)
--> 计次循环首 (10, 宽位置)
--> 计次循环首 (10, 高位置)
--- 如果 (高位置 % 2 = 1)
    隐藏画板.复制 (隐藏画板.宽度 ÷ 10 × (宽位置 - 1), 隐藏画板.高度 ÷
    10 × (高位置 - 1), 棋盘宽度, 隐藏画板.高度 ÷ 10, 演示画板, 演示
    画板.宽度 ÷ 10 × (宽位置 - 1), 演示画板.高度 ÷ 10 × (高位置 -
    1), )
    隐藏画板.复制 (隐藏画板.宽度 ÷ 10 × (宽位置 - 1) + 隐藏画板.宽度 ÷
    20, 隐藏画板.高度 ÷ 10 × (高位置 - 1), 棋盘宽度, 隐藏画板.高度
    ÷ 10, 演示画板, 演示画板.宽度 ÷ 10 × (宽位置 - 1) + 演示画板
    .宽度 ÷ 20, 演示画板.高度 ÷ 10 × (高位置 - 1), )
    --- 如果真 (棋盘宽度 > 隐藏画板.宽度 ÷ 20 且 宽位置 = 10)
        隐藏画板.复制 (0, 隐藏画板.高度 ÷ 10 × (高位置 - 1), 棋盘宽度 -
        隐藏画板.宽度 ÷ 20, 隐藏画板.高度 ÷ 10, 演示画板, 0, 演示画
        板.高度 ÷ 10 × (高位置 - 1), )
    --- 计次循环尾 0
--- 计次循环尾 0
棋盘宽度 = 棋盘宽度 + 2
延时 (速度 \ 2)
--- 判断循环尾 0
```

“棋盘宽度”变量用来保存当前棋盘格的宽度，初始值为0（因为此时尚未画出图形）。用嵌套循环建立10×10的棋盘方格，并通过“高位置”变量的奇偶性控制棋盘格的交错位置。每当100个棋盘格绘画遍历完毕，就将棋盘格的宽度加2个单位，重新遍历，形成每个棋盘格从左到右推进的效果。需要了解的是，每次重新遍历，棋盘格图片宽度就会增加2个像素单位，但每个棋盘格的左边位置其实都没有改变。如果打算像看到的视觉效果那样，每次改变棋盘格的左边起始位置，那实现起来就异常烦琐了。



## 8.5 影视

### 8.5.1 影像框组件

易语言的影像框组件主要用作播放 AVI 格式的动画文件,要求使用标准 AVI 格式文件,否则无法播放。此外,使用此组件,AVI 文件伴音将无法播放出来。

#### 1. “文件名”、“播放”属性

只要设置“播放”属性为真,将 AVI 文件完整路径名赋值给“文件名”属性,就可以播放该 AVI 文件了。

实例代码如下:

```
影像框 1.文件名 = "c:\动画 1.avi"
```

#### 2. “居中播放”、“透明背景”、“播放次数”属性

“居中播放”属性指定影像在组件中心播放。

“透明背景”属性指定影像的背景为透明。

“播放次数”属性指定影像的播放次数。如为 -1,则无限次地循环播放。

#### 3. 影像框组件的特点

影像框的用法与图片框相似,不同之处在于:图片框是容器型组件,可以在其中加入其他组件。移动图片框,其中的组件也会随之移动;而影像框则不具备此特性。

### 8.5.2 高级影像框组件

鉴于影像框组件应用的局限性,从易语言 3.5 版开始,增加了高级影像框组件。高级影像框组件支持更多压缩格式的 AVI 文件。

高级影像框组件的使用方法和影像框组件相仿,增加了几个新的属性便于播放控制。

#### 1. “影像文件名”属性

与影像框组件的“文件名”相同,指定要播放的 AVI 文件的完整路径。

#### 2. “播放位置”和“播放速度”属性

“播放位置”属性是整数型的,可以选择: 0、窗口内居中, 1、扩展到窗口, 2、自动调整窗口。相比较之下,影像框的“居中播放”属性同样是控制影像在组件中的位置,功能就显得逊色多了。

高级影像框组件还增加了“播放速度”属性。这个属性的默认值是 1,即保持正常速度。如果参数值为大于 0 的小数,给的数越小,播放的速度就会越快。

#### 3. “当前帧”和“全部帧数”属性

高级影像框组件增加了对帧的操作。“当前帧”属性可以设置当前播放的帧。“全部帧数”属性是只读属性,只能在程序运行时调用,返回当前播放文件的全部帧数。

打开例程“AVI 播放器.e”,程序运行界面如图 8-13 所示。



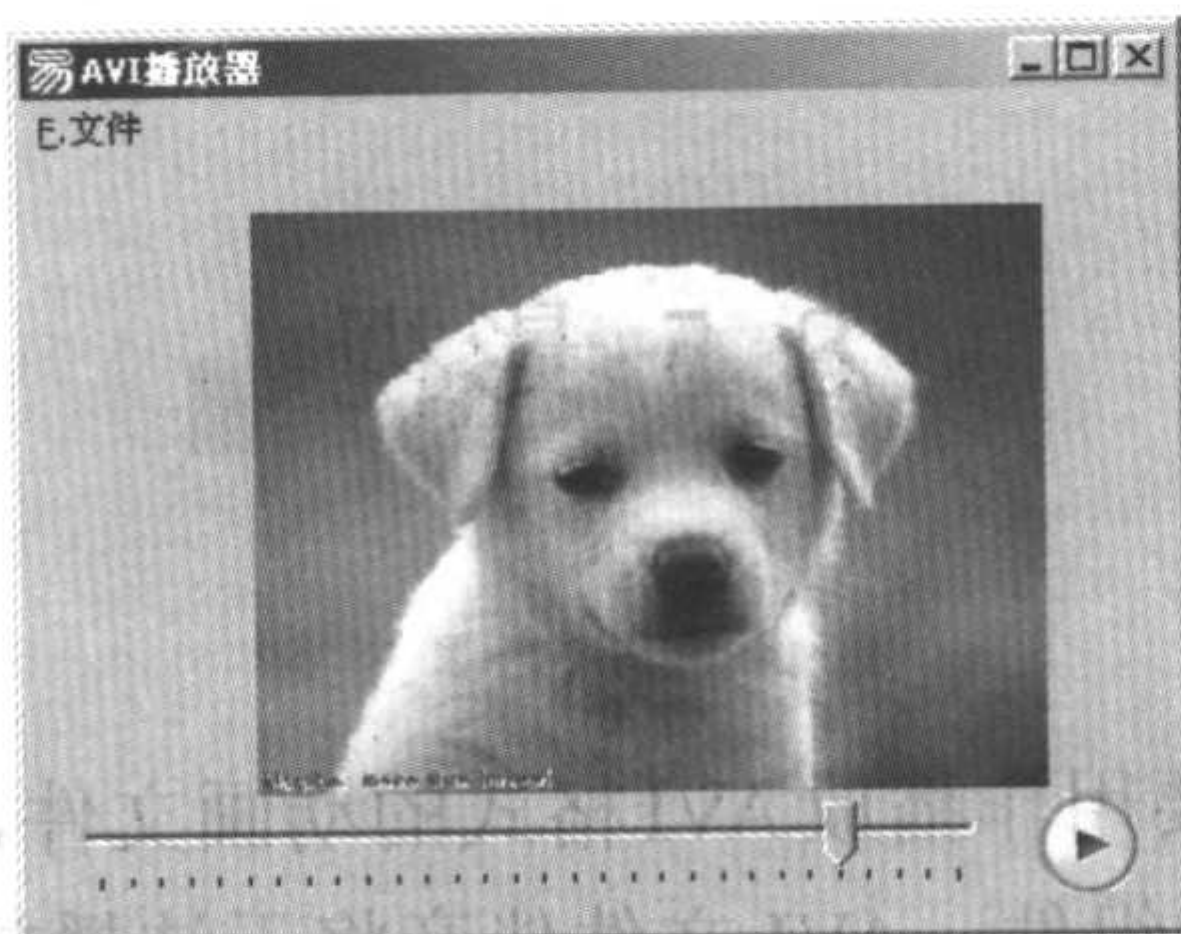


图 8-13 播放 AVI 文件

菜单“\_打开\_被选择”事件子程序代码如下：

```
如果真 (通用对话框1.打开 0)
    高级影像框1.影像文件名 = 通用对话框1.文件名
```

只要“影像文件名”属性中文件路径有效，就可以通过“播放”属性播放或停止播放。可供选择的属性值：0、播放；1、停止；2、暂停；3、继续。

高级影像框有一个特点，就是它所支持的 AVI 文件的格式类型是可以扩充的。只要操作系统安装了相应的文件格式解码器，高级影像框就能够正常的播放该类型格式文件。

如标准的 Windows 操作系统是不支持 MPEG4 压缩格式的，因此高级影像框也不能播放 MPEG4 压缩格式的 AVI 文件。但只要下载安装 MPEG4 格式的解码器后，高级影像框就可以正常播放此类型的 AVI 文件了。

读者可自行在网上搜索并下载安装 MPEG4 解码器（MPEG4 插件），即可让高级影像框支持播放 MPEG4 压缩格式的 AVI 文件了。

## 8.5.3 外部影像组件

1. 用外部 OCX 组件也可以实现视频播放功能。例程“播放电影 2.e”，使用 MediaPlayer 组件播放影像文件。

为了方便，用易语言的组件包装器载入名为“WM 播放器.npk”的汉化文件，要求系统中必须安装有 Windows Media Player 7.0 以上版本，或操作系统安装目录下的 System 目录（Windows NT 操作系统下是 System32 目录）中存在 msdxm.ocx 文件。组件的使用方法与高级影像框组件相仿，其功能更为强大和完善，使用起来却异常简单。程序运行界面如图 8-14 所示。



图 8-14 使用 MediaPlayer 组件



Windows Media Playe 组件同样可以通过安装其他解码器来扩展所支持的文件格式。如安装 rm 解码器“RealoneED.exe”就可以播放 RM 文件了。

组件命令如图 8-15 所示。

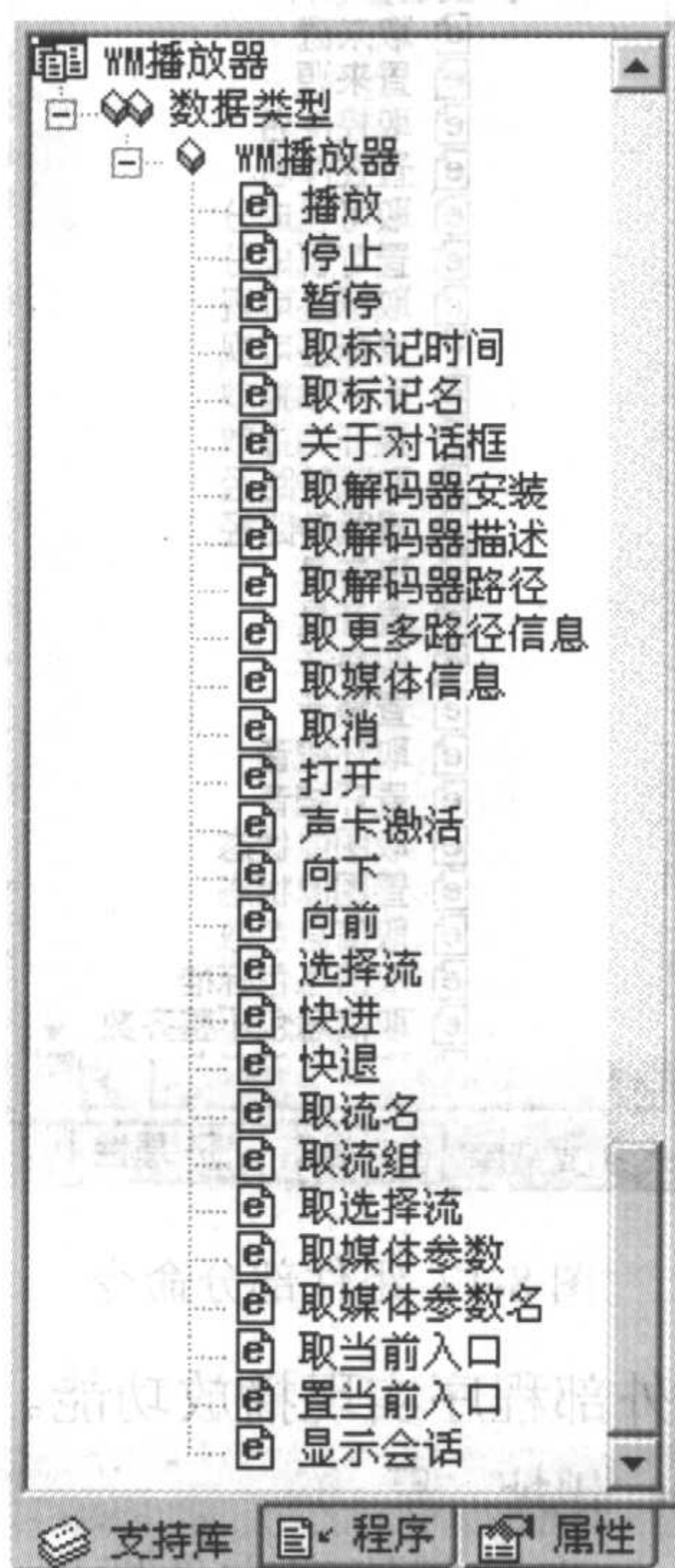


图 8-15 组件命令

用这个组件可以实现更强大的播放功能。这里就不多讲了。

3. RM 文件可以通过另一个 OCX 组件直接播放。例程“RM 播放组件应用示例.e”。首先运行“工具”→“封装类型库与 OCX 组件”，使用“注册组件”功能注册“packcom.dll”外部组件，安装“RM 播放器.npk”汉化文件。程序运行界面如图 8-16 所示。

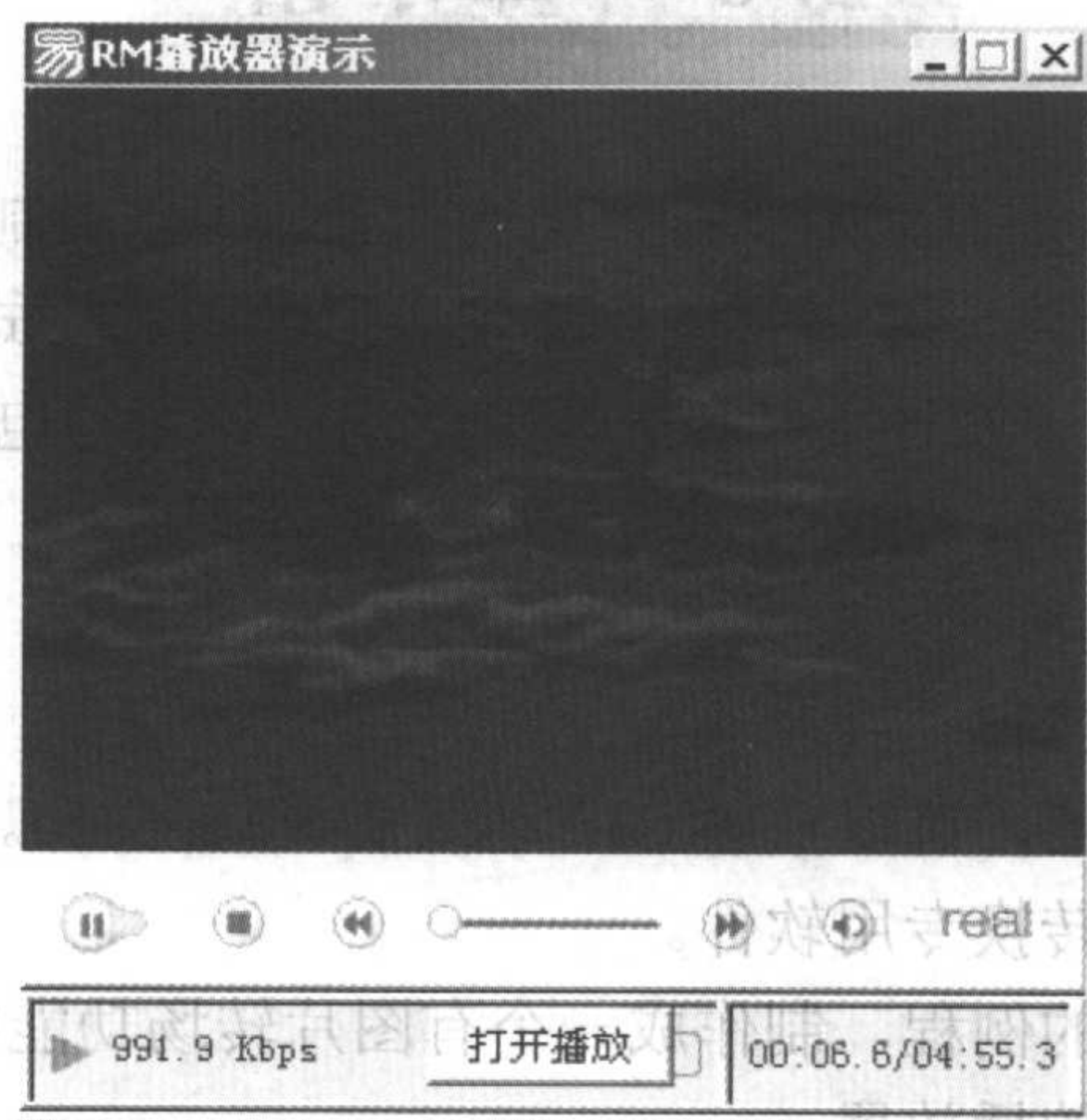


图 8-16 RM 播放器





组件部分命令如图 8-17 所示。

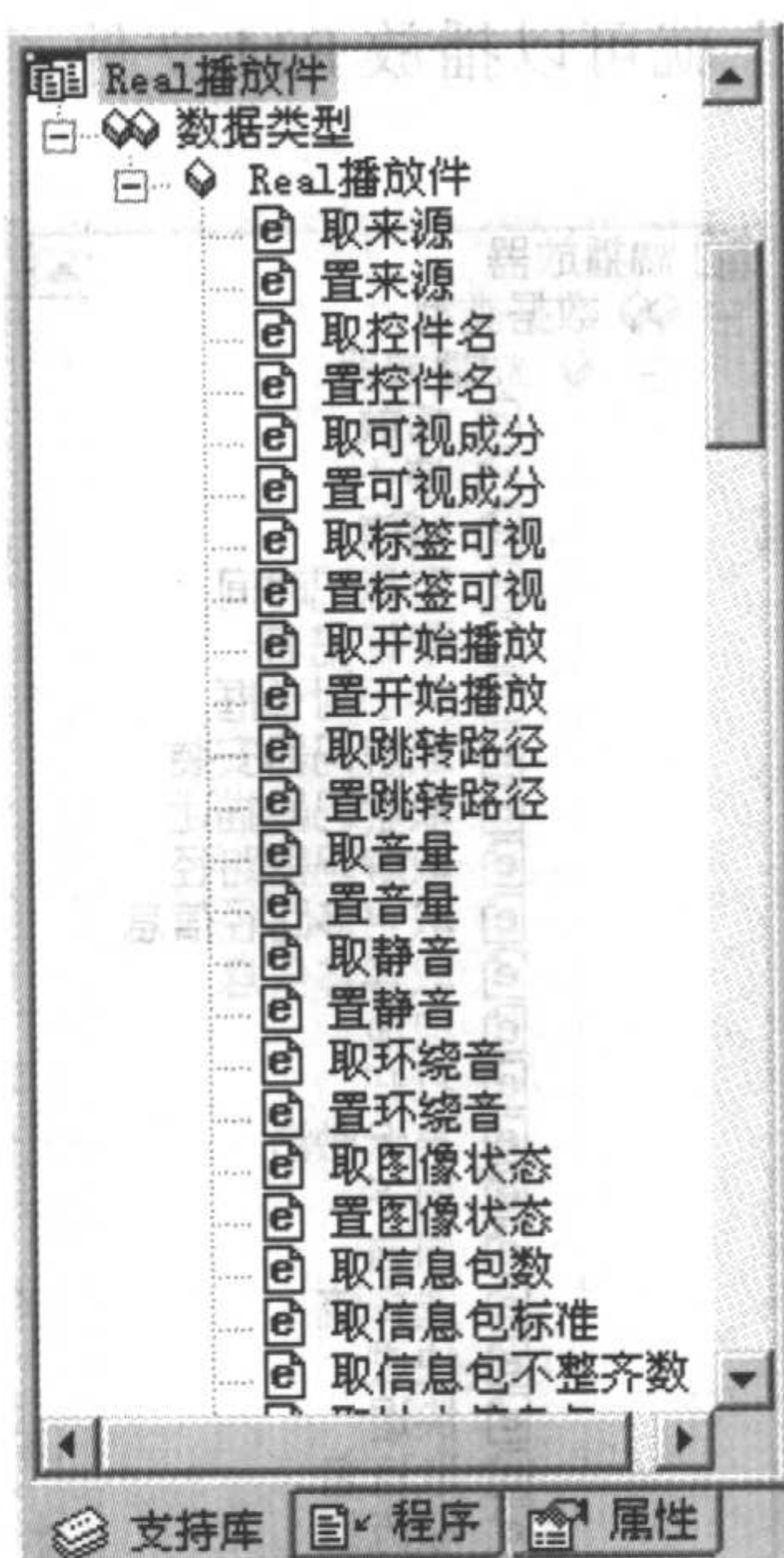


图 8-17 组件部分命令

2. 例程“网络电视.e”利用外部程序实现播放功能。其主要的代码只有一行，如下。

```
运行 (“Explorer.exe” + 地址, 假, )
```

通过调用外部程序 Explorer.exe 来访问影音文件地址，并借助 Explorer.exe 的自动关联文件的特性调用相应的播放器或浏览器播放文件内容。这种实现方法是一种变通的解决办法，减少了程序开发的复杂程序，但同时也降低了程序界面风格的统一性。

## 8.6 本章小结

本章所讲的多媒体应用是大家比较感兴趣的部分，如自己制作一个 MP3 播放器，制作一个播放视频的播放器。当然，易语言还提供了专用的图像格式转换支持库，以供大家进行图像格式转换之用。还有多媒体支持库、文本语音支持库与电话语音支持库，功能较多，因此将在专门的章节中作介绍，在此大家也可以自己学习。

大家还可以进行以下的练习：

1. 制作一个 MP3 播放器。
2. 制作一个动画播放器，可以播放 GIF 动画及 SWF 文件。
3. 制作一个图像格式转换专用软件。
4. 修改本书图片转场的例程，制作成一个有图片转场功能的图片浏览器。
5. 制作播放影像文件的播放器。
6. 配合易语言支持库中的例程自学多媒体支持库、文本语音支持库与电话语音支持库。



## 第九章 网络与通讯

网络的建设促进了计算机技术的发展,特别是个人电脑的普及,让众多计算机用户能便捷地共享信息资源,同时使不同计算机之间也能互相传递信息进行交流。

对于常见的网络程序来说,可以简单的划分为三种类型:网络应用型、数据通讯型和硬件通信型。严格地说,所有的网络应用都是建立在硬件通信和数据通讯的基础之上的,但为更好的了解易语言的网络功能和网络操作命令,本章把基于互联网标准通讯协议程序:即基于 HTTP、FTP、SMTP 等协议的程序,统称为网络应用型程序;把基于自定义数据通讯协议的程序:如聊天室、网络监控、网络游戏等程序,统称为数据通讯型程序;把直接操作硬件端口程序:操作串口/并口的程序,统称为硬件通信型程序。

### 9.1 基础知识

#### 基本概念:

**TCP/IP 协议:** TCP/IP 协议是传输控制协议(TCP)和网际协议(IP)的简称,由于 TCP/IP 是保证数据完整传输的两个基本的重要协议,在网络中具有极为重要的作用。人们通常也把以这两个协议为基础的一族协议称为 TCP/IP 协议族,它包括上百个各种功能的协议,如:远程登录、文件传输和电子邮件等。

**IP 地址:** Internet 上的每一台计算机都被赋予一个世界上惟一的 32 位 Internet 地址(Internet Protocol Address, 简称 IP Address),这一地址可用于与该计算机有关的全部通信。为了方便起见,在应用上一般以 8bit 为一单位,组成四组十进制数字来表示每一台主机的位置。

一般的 IP 地址由 4 组数字组成,每组数字介于 0-255 之间,如某一台电脑的 IP 地址可为: 202.206.65.115,但不能为 202.206.259.3。

**域名地址:** 数字型的 IP 地址不便于用户记忆,于是人们又发明了另一套字符型的地址方案即所谓的域名地址,例如:新浪的网址为“www.sina.com.cn”。每个域名都有一个惟一的 IP 地址与之对应,例如新浪的 IP 为“61.135.152.70”,相应的域名地址信息存放在一个叫域名服务器(DNS, Domain Name Server)的主机内,使用者只需了解易记的域名地址,其对应转换工作就留给了域名服务器 DNS。

**HTML:** Hypertext Marked Language,即超文本标记语言,是一种用来制作超文本文档的简单标记语言。用 HTML 编写的超文本文档称为 HTML 文档,它能独立于各种操作系统平台(如 UNIX, WINDOWS 等)。表 9-1 还列出一些其他常见的标准网络通讯协议,这里不再一





一赘述。

协议名称	应用范围
Telnet	提供远程登录（终端仿真）服务协议
FTP	应用级文件传输服务协议
SMTP/POP3	电子邮件传输协议
SNTP	简单网络管理协议
DNS	域名解析服务，也就是将域名映射成 IP 地址的协议
HTTP	超文本传输协议

表 9-1 常见标准网络通讯协议

## 9.2 网络应用型程序

本章把基于互联网标准通讯协议的程序统称为网络应用型程序。目前已有上百种标准协议，限于篇幅，这里仅对 HTTP、SMTP 和 FTP 协议在易语言中的应用做简单讲解。如果想了解其他协议的应用方法，请查阅相关书籍或资料。

### 9.2.1 网络组件

#### 超级链接框组件



超级链接框可以在程序中实现网页效果的超链接标签：当鼠标移动到超链接地址上时会自动变成手型，同时地址文本颜色发生变化。点击该超链接地址，会根据其属性设置自动选择调用缺省浏览器浏览网页地址，或调用邮件客户端进行编辑邮件工作，并且改变超链接文本颜色表示该地址已访问过。如图 9-1 所示。

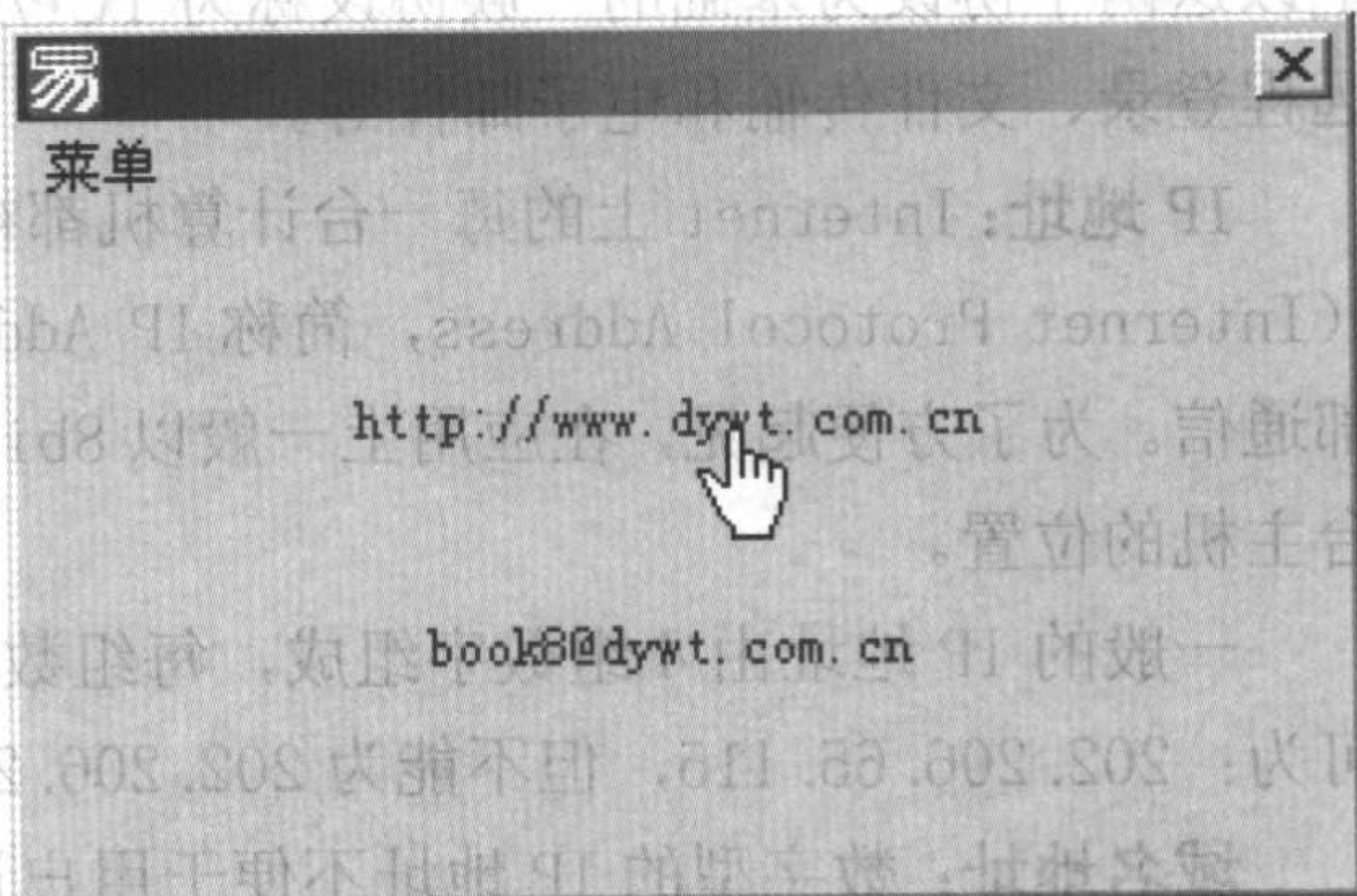


图 9-1 超级链接框

使用超级链接框组件，一般只需要设置其“类型”属性、“电子信箱地址”属性或“Internet 地址”属性即可。

“类型”属性为整数型，有两个可选值：0. 电子信箱地址、1. Internet 地址，默认为 0。当“类型”值为 0 时，“电子信箱地址”属性有效；当“类型”值为 1 时，“Internet 地址”属性有效。



若“标题”属性文本为空，则会自动以“电子信箱地址”属性或“Internet 地址”属性的文本作为显示标题。

利用超级链接框组件来做一个简单的程序，实现该例程不需要写任何代码，只要对组件属性进行相应设置即可，步骤如下：

1. 新建易程序，并在启动窗口中放上两个超级链接框组件；
2. 第一个超级链接框组件的“类型”属性设为“1. Internet 地址”，并在“Internet 地址”属性栏中填入“http://www.dywt.com.cn”；
3. 另一个超级链接框组件的“类型”属性设为“0. 电子信箱地址”，并在“电子信箱地址”属性栏中填入“book8@dywt.com.cn”；

按 F5 运行该程序，就可以看到最终效果了。

当然，上述效果同样可以用程序代码来实现，代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

超级链接框1. 类型 = 1

超级链接框1. Internet地址 = “http://www.dywt.com.cn”

超级链接框2. 类型 = 0

超级链接框2. 电子信箱地址 = “book8@dywt.com.cn”

通过对超级链接框的“边框”、“文本颜色”、“热点颜色”、“访问后的颜色”属性进行设置，可以实现不同效果的超链接标签，甚至是按钮效果。

超级链接框的“跳转()”方法可以实现程序代码控制组件被激活，而不需要等待鼠标点击。这样，大家可以将组件“可视”属性设为假，然后用其他组件事件来调用“跳转”方法，达到特殊效果。比如通过菜单直接访问网页的例子，如图 9-2 所示。



图 9-2 隐藏的超级链接框特殊用法

菜单子程序代码如下：





子程序名	返回值类型	公开	备注
_访问易语言网站_被选择			

隐藏的超级链接框.类型 = 1

隐藏的超级链接框.Internet地址 = "www.dywt.com.cn"

隐藏的超级链接框.跳转 ()

子程序名	返回值类型	公开	备注
_访问邮箱_被选择			

隐藏的超级链接框.类型 = 0

隐藏的超级链接框.电子信箱地址 = "book8@dywt.com.cn"

隐藏的超级链接框.跳转 ()

## 超文本浏览框组件



超文本浏览框提供对标准 HTML 页面的浏览支持,如本地 HTML 文件或 CHM 文件、远程静态或动态网页等。使用此组件可以快捷地开发出小型浏览器、FLASH 播放器、网站专用工具等极有特色的应用程序。

首先大家打开查看随书光盘本章目录中的“易浏览器.e”,运行效果如图 9-3 所示。



图 9-3 易浏览器

该例程使用非常简洁的代码就实现了浏览器的所有基本功能。菜单与工具条的设计及图片组的处理请参阅相关资料,请重点关注对超文本浏览框组件“**执行命令()**”方法的调用及其各种内部事件响应的处理。

### 1. “执行命令()”方法

调用格式: 〈无返回值〉 对象. 执行命令 (欲执行的命令)



执行指定的浏览器命令。

参数“欲执行的命令”类型为“整数型 (int)”。指定欲执行命令的类型，为以下常量值之一：0: #前进；1: #后退；2: #到首页；3: #到搜索页；4: #刷新；5: #停止；6: #另存为；7: #打印；8: #打印预览；9: #页面设置。

例程中相关程序代码为：

子程序名	返回值类型	公开	备注		
_工具条1_被单击					
参数名	类型	参考	可空	数组	备注
按钮索引	整数型				

```

--- 判断 (按钮索引 = 0)
--- 超文本浏览框1. 执行命令 (#到首页)
--- 判断 (按钮索引 = 1)
--- 超文本浏览框1. 执行命令 (#后退)
--- 判断 (按钮索引 = 2)
--- 超文本浏览框1. 执行命令 (#前进)
--- 判断 (按钮索引 = 3)
--- 超文本浏览框1. 执行命令 (#停止)
--- 判断 (按钮索引 = 4)
--- 超文本浏览框1. 执行命令 (#刷新)
--- 判断 (按钮索引 = 6)
--- 选择字体大小 (2)
--- 判断 (按钮索引 = 7)
--- 超文本浏览框1. 执行命令 (#打印)

```

## 2. “即将跳转”事件

在浏览器即将跳转到另一个页面之前产生此事件，在事件处理子程序中读取“地址”属性即可得知即将跳转到的地址，返回值为“假”时不允许跳转，返回值为“真”或不返回值则允许跳转。

例程中相关代码为：

子程序名	返回值类型	公开	备注
_超文本浏览框1_即将跳转	逻辑型		

状态条1. 置文本 (1, “正在转向” + #左引号 + 超文本浏览框1. 地址 + #右引号)

## 3. “跳转完毕”事件

当浏览器已跳转到另一个页面之后产生此事件，在事件处理子程序中读取“地址”属性就可得知已跳转到的地址。





## 4. “载入开始”事件

本事件在“即将跳转”事件之后触发，表示浏览器已开始载入将要显示的文档。

## 5. “载入进度改变”事件

在浏览器载入文档的过程中，每当文档被载入一部分即触发本事件，用作通知载入进度。

例程中相关代码为：

子程序名	返回值类型	公开	备注		
超文本浏览框1_载入进度改变					
参数名	类型	参考	可空	数组	备注
进度百分比	整数型				

状态条1.置文本 (1, “已经载入 ” + 到文本 (进度百分比) + “ %” )

## 6. “载入完毕”事件

当将要显示在浏览器内的文档被载入完毕后触发本事件。

## 7. “已就绪”事件

当浏览器已经将所需显示的文档处理完毕后发送本事件，在事件处理子程序中读取“地址”属性就可得到已就绪文档的地址。

例程中相关代码为：

子程序名	返回值类型	公开	备注
超文本浏览框1_已就绪			

状态条1.置文本 (1, “就绪” )

地址编辑框.内容 = 超文本浏览框1.地址

## 8. “状态文本被改变”事件

当浏览器的状态条文本被改变后发送本事件，在事件处理子程序中读取“状态条文本”属性即可得知其内容。

例程中相关代码为：

子程序名	返回值类型	公开	备注
超文本浏览框1_状态文本被改变			

状态条1.置文本 (1, 超文本浏览框1.状态条文本)

## 9. “标题被改变”事件

当浏览器的标题文本被改变后发送本事件，在事件处理子程序中读取“标题”属性即可得知其内容。

例程中相关代码为：



子程序名	返回值类型	公开	备注
_超文本浏览框1_标题被改变			

标题 = 超文本浏览框1. 标题

### 10. “命令状态被改变”事件

当“前进”、“后退”等命令的允许状态被改变后发生此事件，用户程序应该根据状态值允许或禁止对应的按钮或菜单项。

例程中相关代码为：

子程序名	返回值类型	公开	备 注		
_超文本浏览框1_命令状态被改变					
参数名	类 型	参考	可空	数组	备 注
命令	整数型				
是否被允许	逻辑型				

```

--- 如果 (是否被允许 = 假)
    --- 判断 (命令 = #前进)
        工具条1. 加入状态 (2, #禁止)
        前进. 禁止 = 真
    --- 判断 (命令 = #后退)
        工具条1. 加入状态 (1, #禁止)
        后退. 禁止 = 真
    --- 判断 (命令 = #前进)
        工具条1. 去除状态 (2, #禁止)
        前进. 禁止 = 假
    --- 判断 (命令 = #后退)
        工具条1. 去除状态 (1, #禁止)
        后退. 禁止 = 假
    
```

### 11. “即将打开新窗口”事件

在浏览器即将打开新窗口浏览另一个页面之前产生此事件，事件处理子程序返回“假”则不允许打开，返回“真”或不返回值则允许打开。

可以看出，合理的使用组件所提供的各种功能，能够为开发工作提供很大的便利。易浏览器例程的功能在实际应用中尚显单薄，但如果对此例程功能进行扩充，主要是对各种内部事件的响应多加些处理，就能开发出更好更完善的浏览器来。

现在再来看一个 WEB 邮箱登录器的小程序。





很多企业都有自己的企业邮局，它是企业信息化建设的重要组成部分。大家在为企业定制信息化管理系统时，经常被要求把收发邮件功能集成到程序当中。易语言提供邮件发送的相关命令，大家可以很轻松的在程序中集成发信功能，为开发提供了很大的便利。但由于没有接收邮件的相关命令，大多数朋友只能用客户控件实现简单的邮件提醒功能，水平高一点的会自己写解码函数或通过第三方控件来实现收信解码，但总是有些不尽如人意。其实，目前企业邮局大多提供 WEB 方式访问，只要在程序中嵌入 WEB 页面显示，并让其自动登录邮箱首页，就可以实现在程序中集成收发邮件功能了。程序运行效果如图 9-4 所示。

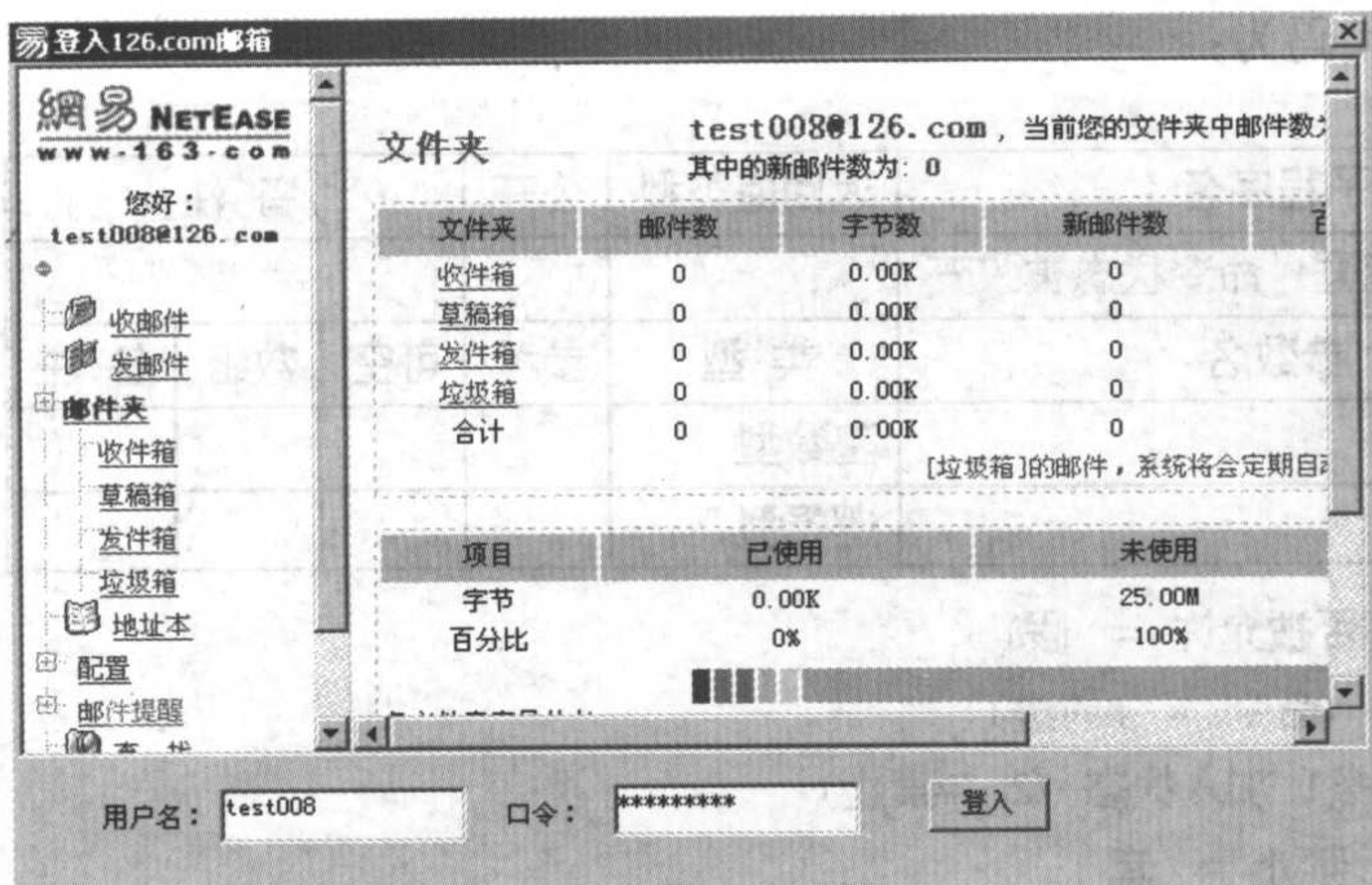


图 9-4 WEB 邮箱登录器

需要解决的关键性技术问题是如何实现自动登录到邮箱首页。这里大家需要先了解一些 HTML 文档相关的基础知识，可参阅其他相关资料。

以 www.126.com 网站 WEB 邮箱为例。

首先分析 www.126.com 网站首页页面的源码。

在 IE 浏览器地址栏输入 www.126.com 网址并进入页面，选择 IE 菜单“查看”→“源代码”，可以看到如图 9-5 所示内容。

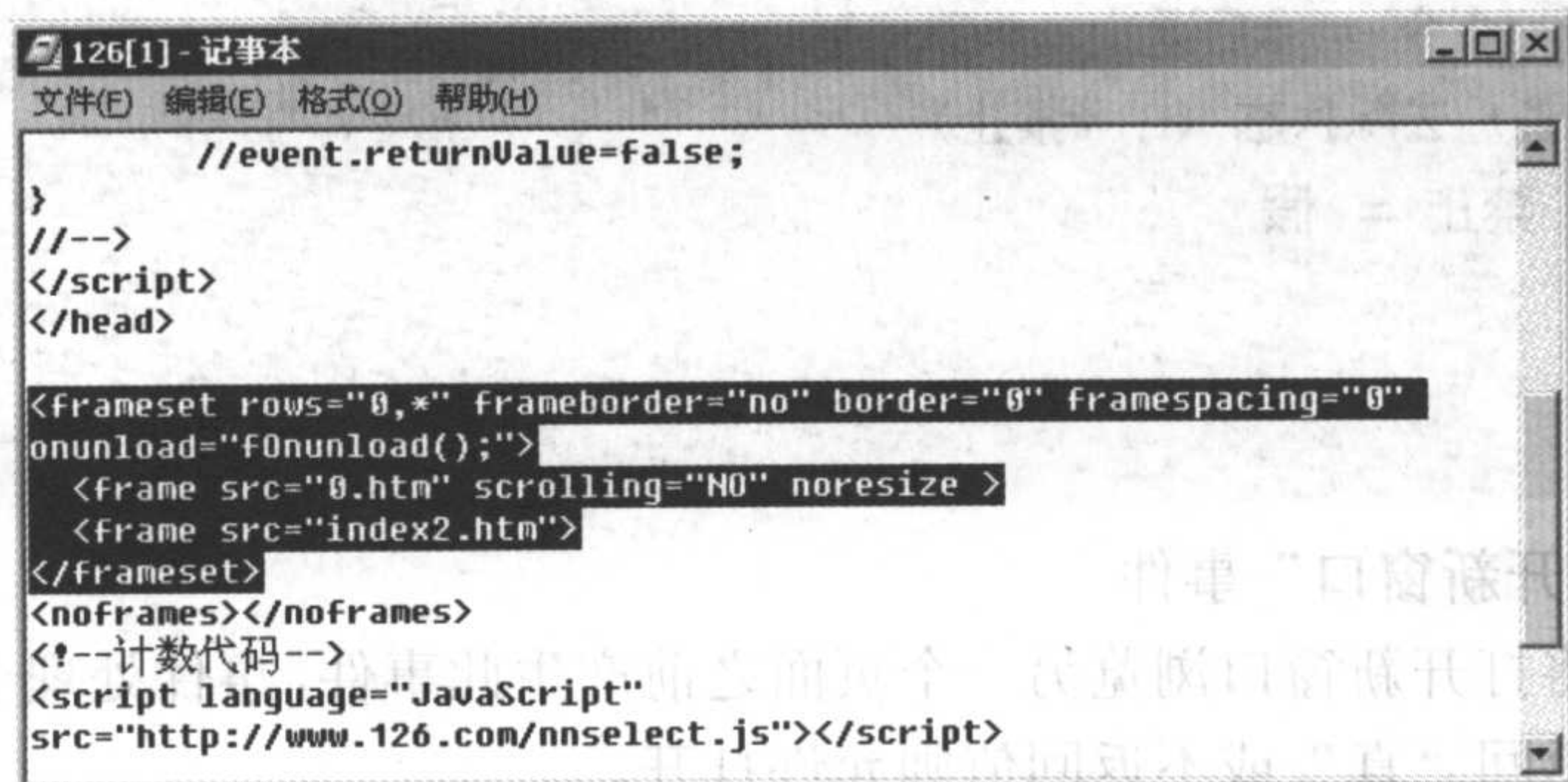


图 9-5 126 首页源代码

反白部分代码说明此页面是个框架页，在浏览器中分别输入：

http://www.126.com/0.htm 和 http://www.126.com/index2.htm 两个地址查看页面结



果,可以发现 index2.htm 是真正的首页面。

现在来看 index2.htm 源代码中登录邮箱提交参数部分的网页代码(注:HTML 语言中提交数据代码段位于<form>与</form>标记中间。为清晰起见,下段代码略做删节):

```
<form method=POST target="_self" action="http://entry.126.com/cgi/login" name="form"
onSubmit="return fCheck()">
  <input name="domain" type=hidden value="126.com">
  <input name="language" type=hidden value="0">
  <input name="bCookie" type=hidden value="">
  .....
  <td height="27" colspan="2" nowrap>用户名:
    <input name="user" type="text"
style="font-size:9pt;font-family:Verdana;border:solid 2 #000000" size="15">
    <span class="redbig">@ 126.com</span></td>
  .....
  <td height="55" nowrap>密 码:
    <input name="pass" type="password"
style="font-size:9pt;font-family:Verdana;border:solid 2 #000000" size="15">
    <br>
    <a href="http://entry.126.com/cgi/reg?funcid=rpstart">忘记密码? </a>
  <br></td>
  <td width="144">
    <input name="enter.x" type="submit" value="登录"
style="height:40;width:80;font-size:13.5pt;padding-top:3;font-family:黑体;border-left:solid
2 #444444;border-top:solid 2 #444444;border-right:solid 3 #000000;border-bottom:solid 3
#000000;background:#D4D0C8;cursor:hand;" onMouseOver="style.background='#e8e6e0'"
onMouseOut="style.background='#D4D0C8'"></td>
  .....
    <select name="style">
    <option value="1">随心 DIY</option>
    <option value="-1">默认风格</option>
    <option value="0">简约风格</option>
  .....
</form>
```

看起来似乎比较杂乱,可以把它简化一下,只保留所需要的内容:

```
<form method=POST target="_self" action="http://entry.126.com/cgi/login"
name="form" onSubmit="return fCheck()">
  <input name="domain" ..... value="126.com">
  <input name="language" ..... value="0">
```





```
<input name="bCookie" ..... value="">  
<input name="user" .....>  
<input name="pass" .....>  
<input name="enter.x" ..... value="登录" .....>  
<select name="style"> “1”随心 DIY “-1”默认风格 “0”简约风格  
</form>
```

上述代码可描述为：“登录”按钮被按下时，对 domain、language、bCookie、user、pass、enter.x、style 六个参数赋值后，经过 fCheck() 函数处理，使用 POST 方法提交至 http://entry.126.com/cgi/login 页面。

因为源代码中 fCheck() 函数没有对这些参数值进行改变，为简化讲解，不做说明。但该函数调用了一个 fStyle() 函数，变动了访问页面地址，且增加了一个 verifycookie=1 参数：http://entry.126.com/cgi/login?verifycookie=1&language=0&style=-1。

那么，根据以上信息，可以写出带参数访问页面的 URL 为：

```
http://entry.126.com/cgi/login?verifycookie=1&language=0&style=-1&domain=126.com&  
bCookie=& user="+用户名+"& pass="+密码+"& enter.x= %B5%C7%C2%BC
```

(注：%B5%C7%C2%BC 是汉字“登录”的内码)

把自己的用户名和密码代入 URL，在浏览器地址栏中输入该地址，成功。

现在来根据分析结果编写程序。

新建一个易程序，程序界面如图 9-6 所示。

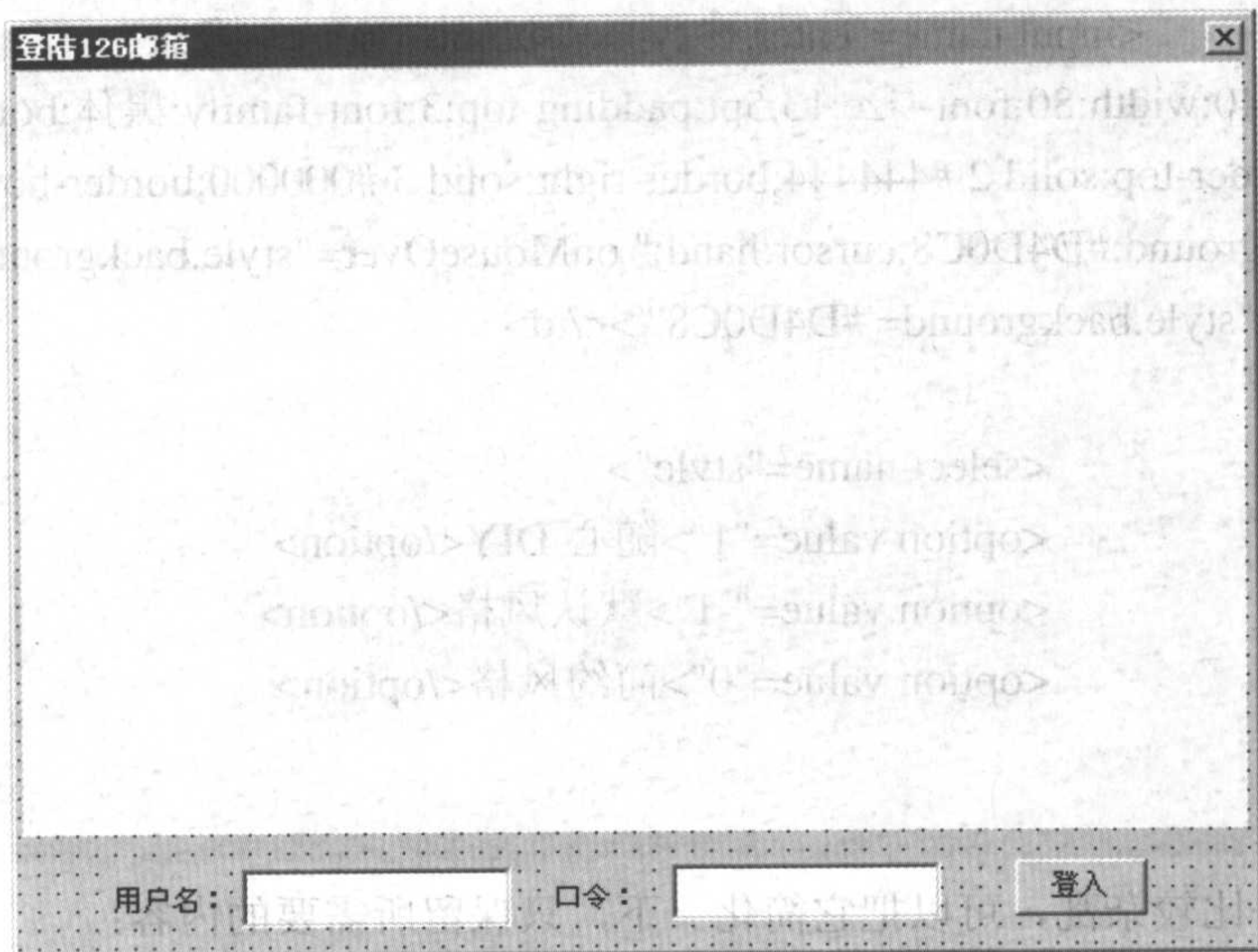


图 9-6 登录邮箱程序界面

程序中，用来输入密码的“编辑框 2”的“输入方式”属性为“2. 密码输入”。

在“\_按钮 1\_被单击()”事件中加入一句代码：



子程序名	返回值类型	公开	备注
_按钮1_被单击			

超文本浏览框1.地址 = "http://entry.126.com/cgi/login?verifycookie=1&language=0&style=1&domain=126.com&bCookie=@user=" + 编辑框1.内容 + "&pass=" + 编辑框2.内容 + "&enter.x=%B5%C7%C2%BC"

至此，这个 126 的自动登录器就完成了。运行效果同前面的图 9-4 所示。

因为动态页面在处理时，并不一定要求每个参数都必须传递，本实例中的访问地址其实只需要传递用户名和密码即可完成登录，大家可以自己试试。

"http://entry.126.com/cgi/login?user="+用户名+"&pass="+密码

**注意：**本例中并没有使用页面要求的 POST 方法提交参数，而是使用的 GET 方法——将参数放在 URL 中一并提交给服务器。并不是所有网站都能支持 GET 方法与 POST 方法混用，当遇到这种情况的时候，只能调用 API 实现数据 POST 方法。另外，使用 GET 方法提交参数时，把用户名和密码都暴露在 URL 中，不利于密码保护，这一点必须注意。

### 9.2.2 互联网支持库

互联网支持库用来扩展易语言对网络访问操作的能力，目前 1.0 版提供“邮件发送”、“HTTP 及 FTP 操作”和“拨号上网”三个类别，可以满足开发网络程序的基本需要。

#### 拨号上网

拨号连接是指通过调制解调器设备连接到网络或 Internet。通常可能有一或两个拨号连接，是与 Internet 或是与虚拟网络的连接。“配置连接”方法请参考 Windows 帮助和支持。

打开例程“拨号上网.e”，并运行。只要系统中设置过连接，配置文件名就会被加入到组合框中。如图 9-6 所示。

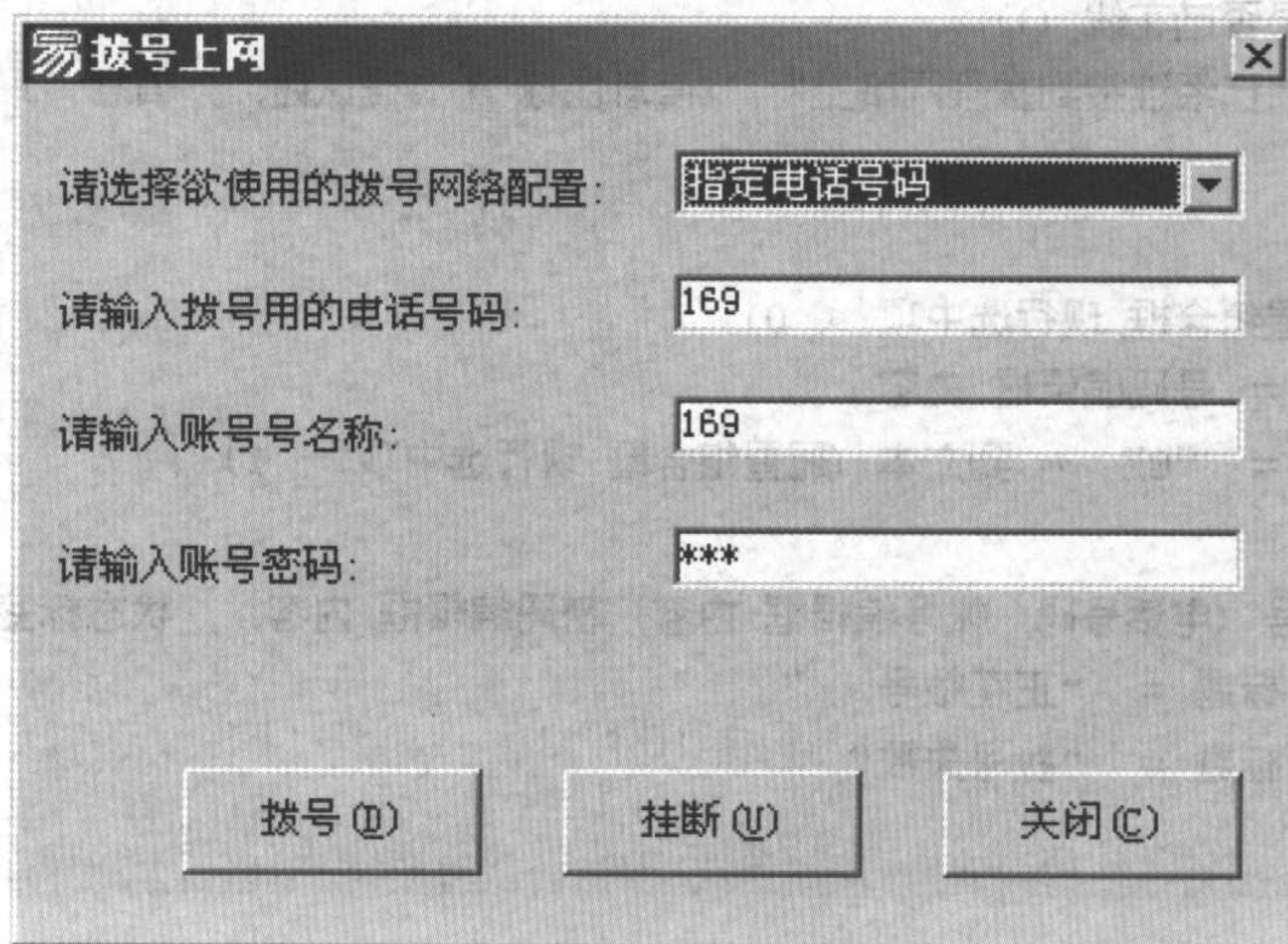


图 9-6 取连接名称





相关程序代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

变量名	类型	静态	数组	备注
拨号连接数	整数型			
计次变量	整数型			

配置组合框.加入项目 (“指定电话号码”,)

拨号连接数 = 取拨号连接数 ()

--- 计次循环首 (拨号连接数, 计次变量)

配置组合框.加入项目 (取连接名称 (计次变量 - 1), )

--- 计次循环尾 ()

配置组合框.现行选中项 = 0

上述代码中使用“取拨号连接数 ()”取得系统中网络配置文件的数量，并用一个计次循环把取得的连接名称加入到组合框中。

“取连接名称 ()”按拨号网络配置文件的索引获取网络配置信息，索引值从 0 开始，最大值为“取拨号连接数 ()”命令的返回值减一。

如果所选择网络配置文件正确，通过单击“拨号按钮”就可以连接到因特网上。相关程序代码如下：

子程序名	返回值类型	公开	备注
_拨号按钮_被单击			

变量名	类型	静态	数组	备注
电话号码	文本型			

--- 如果真 (是否已在线 ())

信息框 (“已经连接到拨号网络上”, #信息图标 + #确认钮, “信息”)

返回 ()

--- 如果 (配置组合框.现行选中项 ≤ 0)

电话号码 = 号码编辑框.内容

--- 电话号码 = “@” + 到文本 (配置组合框.现行选中项 - 1)

--- 如果 (拨号 (电话号码, 帐号编辑框.内容, 密码编辑框.内容, , 状态标签))

状态标签.标题 = “正在拨号...”

--- 状态标签.标题 = “拨号失败”

## 邮件发送

使用易语言提供的邮件发送命令可以编写发送邮件程序。运行例程“易邮件发送器.e”。



运行效果如图 9-7 所示。

图 9-7 易邮件发送器

先来关注一下例程中所用到的“邮件发送”类别中几个重要的命令函数。

“**连接发信服务器()**”命令连接到指定的 SMTP 邮件发送服务器，成功返回“真”，失败返回“假”。该命令支持身份验证和超时判断。每次成功连接发信服务器，完成发信操作后，要及时调用“**断开发信服务器()**”命令释放网络资源。实例代码如下：

```

如果真 (连接发信服务器 (邮件服务器地址, 内容, 到数值 (邮件服务器端口, 内容), 用
  户名, 内容, 密码, 内容, ) = 假)
  状态标签.标题 = “连接邮件服务器失败!”
  返回 0

```

**注意：**如果连接邮件服务器失败，是无法从返回值中了解到连接失败原因的。在实际开发工作中，可以用“客户”组件或“网络客户端”数据类型预先连接服务器地址端口，检测是否有效，这样就可以辅助判断连接失败是网络故障还是用户名、密码错误，不过会耗费多一些时间。

“**断开发信服务器()**”命令断开通过“**连接发信服务器()**”命令所建立的与 SMTP 邮件发送服务器的连接。每次成功连接后可以发送多封邮件，而不必每发送一封邮件就断开连接。

“**添加附件文件()**”命令添加指定的文件附件到即将发送的邮件中，成功返回“真”，失败返回“假”。需要提供的参数是文件的完全路径。程序代码如下：





子程序名	返回值类型	公开	备注
_添加附件_被单击			

```
--- 如果真 (通用对话框1. 打开 () = 真)
    --- 如果 (添加附件文件 (通用对话框1. 文件名) = 真)
        当前附件数目 = 当前附件数目 + 1
        附件标签. 标题 = 到文本 (当前附件数目)
        信息框 ("添加附件失败!", #错误图标, "失败")
    退出子程序
```

“添加附件数据 ()”命令添加附件数据到即将发送的邮件中，成功返回“真”，失败返回“假”。和“添加附件文件 ()”命令不同的是，本命令不要求附件以文件形式存在，如动态处理后的图片数据。实例代码如下：

添加附件数据 (字节集容器, 附件标题)

或

添加附件数据 (读入文件 (文件的完全路径), 附件标题)

“清除所有附件 ()”命令清除所有已添加到即将发送邮件中的附件数据。

“发送邮件 ()”命令发送邮件到指定信箱，注意邮件中包含所有使用“添加附件文件 ()”和“添加附件数据 ()”命令添加的附件。成功返回空文本，失败返回具体错误信息文本。程序代码如下：

```
发送结果 = 发送邮件 (邮件标题. 内容, 邮件内容. 内容, 发向. 内容, 抄送. 内容, 暗送. 内容, 回复地址. 内容, )
```

```
--- 如果 (发送结果 = "")
```

```
    状态标签. 标题 = "发送邮件成功!"
```

```
    状态标签. 标题 = 发送结果
```

```
    断开发信服务器 ()
```

实际应用中，连接一次邮件服务器后可以发送多份邮件，不必每次都断开。

关于本程序的使用，有以下三点需要注意：

如果“回复地址”不填，有可能会发送失败（回复地址一般就是发信人地址）；

如果上一次发送了带附件的邮件，再接着发送新的邮件时，应“清除附件”，否则上次添加的附件会再次被发送；

邮件标题可以不填，但建议填写完整。

当然也可以通过修改程序来解决以上问题，大家可以自己完善这个程序。

## FTP 操作

HTTP 和 FTP 是两种网络协议，其功能比较接近，HTTP 用于超文本传输，FTP 用于网络文件传输。

首先看一下例程“易之 FTP.e”，了解 FTP 相关命令操作。

此例程利用支持库相关命令，连接 FTP 服务器和操作 FTP 服务器上的文件。运行效果如图 9-8 所示。



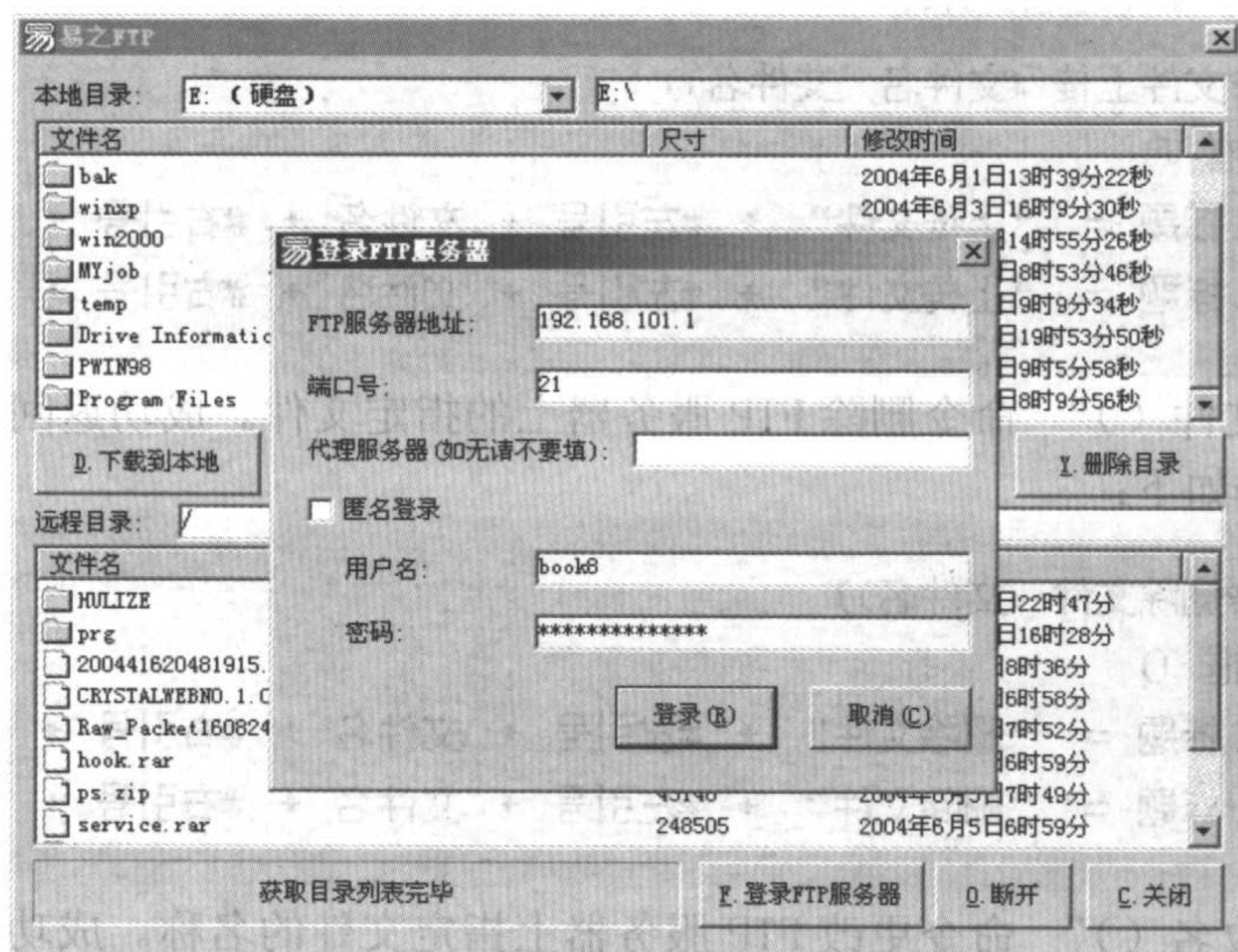


图 9-8 易之 FTP

登录窗口用来连接 FTP 服务器。“置代理服务器 ( )” 命令设置 HTTP 及 FTP 操作中所使用代理服务器的地址，可以为服务器 IP 地址或服务器名称。

“连接 FTP 服务器 ( )” 命令返回逻辑值，成功为“真”。每次 FTP 操作完毕后要调用“断开 FTP 服务器 ( )” 命令断开所建立的连接，释放网络资源。程序代码如下：

置代理服务器 (代理服务器编辑框. 内容)

如果 (匿名登录选择框. 选中 = 真)

登录成功 = 连接FTP服务器 (地址编辑框. 内容, , 到数值 (端口编辑框. 内容), )

登录成功 = 连接FTP服务器 (地址编辑框. 内容, 用户名编辑框. 内容, 密码编辑框. 内容, 到数值 (端口编辑框. 内容), )

“断开 FTP 服务器 ( )” 命令断开 FTP 服务器的连接，本命令执行后不能再进行 FTP 操作。

“FTP 文件下载 ( )” 命令将 FTP 服务器上的指定文件下载到本地，成功返回“真”，失败返回“假”。程序代码如下：

如果 (FTP文件下载 (文件名, 文件名))

填充本地框 ( )

状态标签. 标题 = “下载文件” + #左引号 + 文件名 + #右引号 + “成功”

状态标签. 标题 = “下载文件” + #左引号 + 文件名 + #右引号 + “失败”

“FTP 文件上传 ( )” 命令将本地文件上传到 FTP 服务器上的指定位置，成功返回“真”，失败返回“假”。程序代码如下：





```
-- 如果 (FTP文件上传 (文件名, 文件名))
    填充远程框 ()
    状态标签.标题 = "上传文件" + #左引号 + 文件名 + #右引号 + "成功"
    状态标签.标题 = "上传文件" + #左引号 + 文件名 + #右引号 + "失败"
```

“FTP 删除文件 ()” 命令删除 FTP 服务器上的指定文件。成功返回“真”，失败返回“假”。程序代码如下：

```
-- 如果 (FTP删除文件 (文件名))
    填充远程框 ()
    状态标签.标题 = "删除文件" + #左引号 + 文件名 + #右引号 + "成功"
    状态标签.标题 = "删除文件" + #左引号 + 文件名 + #右引号 + "失败"
```

“FTP 文件改名 ()” 命令更改 FTP 服务器上指定文件的名称。成功返回“真”，失败返回“假”。程序代码如下：

```
-- 如果 (FTP文件改名 (原有文件名, 远程文件框, 结束编辑文本))
    状态标签.标题 = 变量1 + "成功"
    返回 (真)
    状态标签.标题 = 变量1 + "失败"
    返回 (假)
```

“FTP 创建目录 ()” 命令在 FTP 服务器上指定位置处创建新的目录。成功返回“真”，失败返回“假”。程序代码如下：

```
-- 如果 (FTP创建目录 (目录名称))
    填充远程框 ()
    状态标签.标题 = "创建新目录" + #左引号 + 目录名称 + #右引号 + "成功"
    状态标签.标题 = "创建新目录" + #左引号 + 目录名称 + #右引号 + "失败"
```

“FTP 删除目录 ()” 命令删除 FTP 服务器上的指定目录。成功返回“真”，失败返回“假”。程序代码如下：

```
-- 如果 (FTP删除目录 (目录名))
    填充远程框 ()
    状态标签.标题 = "删除目录" + #左引号 + 目录名 + #右引号 + "成功"
    状态标签.标题 = "删除目录" + #左引号 + 目录名 + #右引号 + "失败"
```

“FTP 置现行目录 ()” 命令设置 FTP 服务器上的当前目录，设置后可以在其他 FTP 命令中使用相对路径来指定文件。成功返回“真”，失败返回“假”。程序代码如下：

```
FTP置现行目录 ("\\")
'将远端目录调整到根目录
```



“FTP 目录列表 ()” 命令返回 FTP 服务器上指定目录内的所有匹配文件和子目录信息。成功返回被找到的文件和子目录的数目，失败返回 0。程序代码如下：

```
数目 = FTP目录列表 (“*.*”, , 文件名, 文件属性, 文件尺寸, 文件时间)
```

在 FTP 应用中，除了开发标准 FTP 客户端程序外，还经常用于在线更新升级、内部信息化管理公文共享等方面。FTP 相较 HTTP 最大的优势在于其权限管理功能的完备，能够很轻松的控制用户限时限速的对网络文件进行安全的读写。

## HTTP 操作

再来看看 HTTP 方面应用的实例。HTTP 相关标准命令只有两条：“置代理服务器 ()”和“HTTP 读文件 ()”命令。但只要运用合理，同样也能实现较多功能。先看一个从浏览器获取网页图片的例子。运行效果如图 9-9 所示。



图 9-9 获取网页图片

从图中可以看出，在浏览器的右键菜单中增加了“导出图片”菜单项，每当发现自己喜欢的图片后，可以立即将其调入自己的程序中，这称之为“图片即时贴”。

为了便于讲解，这里已尽量简化了例程，但仍免不了牵涉少许注册表手工操作，以及 VBScript 脚本的编写。若读者有兴趣对其进行扩充完善，可以自行查阅其他相关资料。

### 第一步：增加浏览器右键菜单项

点击左下角“开始”→“运行”，输入 Regedit.exe，点击“确定”按钮，打开注册表编辑器。

在 HKEY\_CURRENT\_USER\Software\Microsoft\Internet Explorer\MenuExt\上新建一个项，命名为“导出图片”，此为右键显示菜单名称。

选择此新建项，双击右部“(默认)”参数，在弹出窗口中输入 c:\cmdline.html，这里指定脚本文件路径。

在空白处点右键，新建 DWORD 值，名称为 contexts，修改其数值为 2，代表此右键菜单项只对图片有效。至此，手工添加浏览器右键菜单项工作就完成了。修改结果如图 9-10 所示。



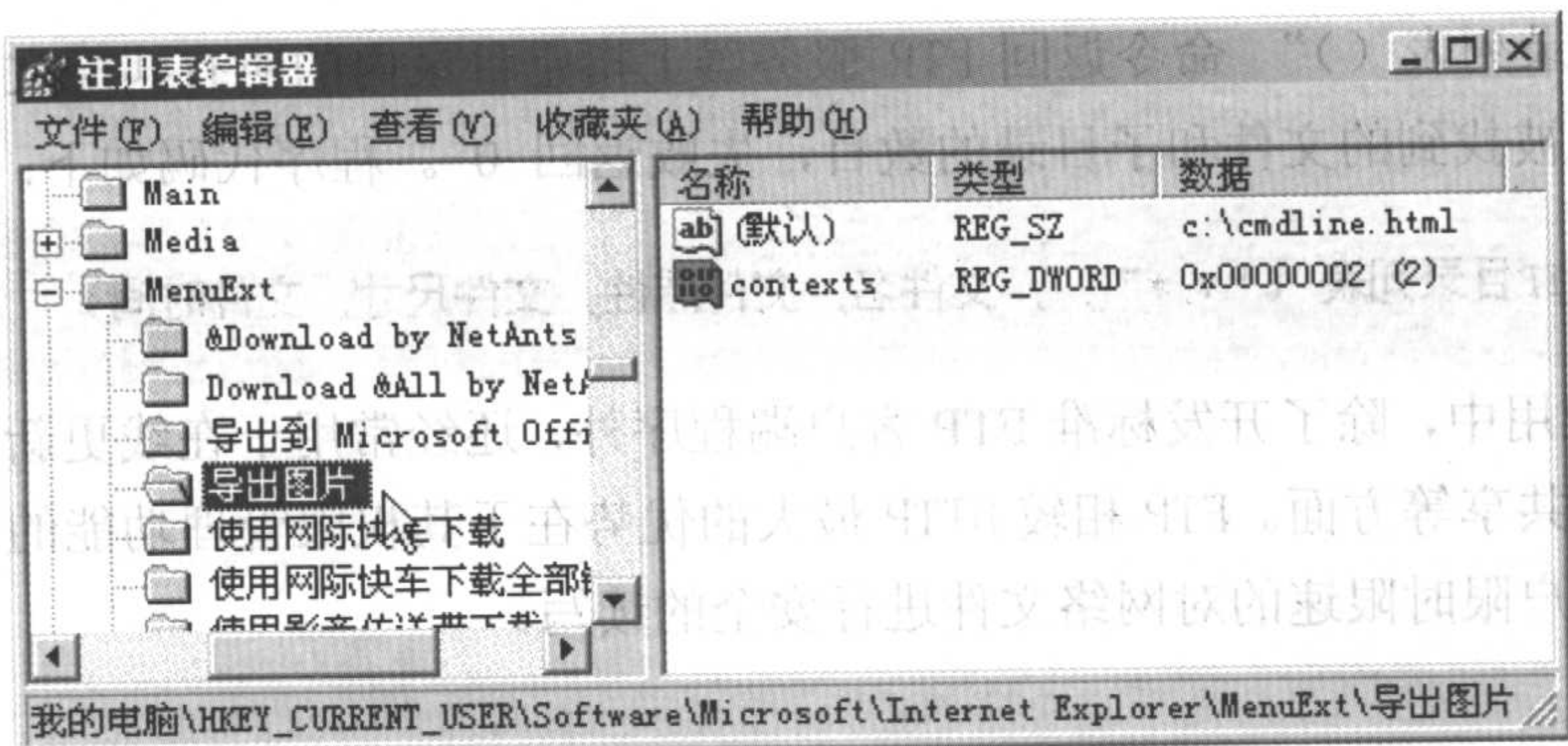


图 9-10 手工修改注册表添加浏览器右键菜单项

## 第二步：创建脚本文件

为实现右键功能，必须调用脚本文件。本例采用 VBScript 编写脚本文件。文件名则为刚才在注册表中设定的 cmdline.html，代码如下：

```
<script language = "VBScript">
sub OnContextMenu()
    set srcEvent = external.menuArguments.event
    set EventElement =
external.menuArguments.document.elementFromPoint(srcEvent.clientX,
srcEvent.clientY)
    On Error Resume Next
    Dim MyObj
    Set MyObj=CreateObject("WScript.Shell")
    if err = 0 then
        MyObj.Run "c:\cmdline.exe " & EventElement.href <!--带参数执行-->
    else
        alert "发生错误!错误号为:"& err
    end if
end sub
call OnContextMenu()
</script>
```

直接用记事本工具将上述代码保存为 c:\cmdline.html 文件，脚本编写工作就全部完成了。

## 第三步：编写程序代码

新建一个易程序，放上一个画板控件，界面设计就算完成了。下面代码是实现窗口根据图片大小自动调整。

全部代码都在“\_启动窗口\_创建完毕”子程序中。具体内容如下：



子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

变量名	类型	静态	数组	备注
地址参数	文本型		0	
图片	字节集			
图片号	整数型			

```

_启动窗口.总在最前 = 真
画板1.自动重画 = 真
取命令行 (地址参数)
' 取回浏览器传递回来的图片地址
    如果真 (取数组成员数 (地址参数) = 0)
        禁止直接运行本程序
        信息框 ("请在浏览器中网页图片上点击右键调用本程序", 0, )
        结束 0
    标题 = 地址参数 [1]
    图片 = HTTP读文件 (地址参数 [1])
    ' 用HTTP读文件 ()命令下载图片文件

    图片号 = 载入图片 (图片)
    画板1.宽度 = 画板1.取图片宽度 (图片号)
    画板1.高度 = 画板1.取图片高度 (图片号)
    _启动窗口.宽度 = 画板1.宽度 + 6
    _启动窗口.高度 = 画板1.高度 + 26
    画板1.移动 (0, 0, 取用户区宽度 (), 取用户区高度 ())
    画板1.画图片 (图片号, 0, 0, , , )
    
```

至此，程序已全部编写完成，接下来的工作就是将此程序进行编译，然后按前面脚本文件中标明的，保存为 c:\cmdline.exe。

现在，打开一个浏览器窗口，访问任意一个网站，在任何一张图片上点击右键，会发现已经多了一项“导出图片”项。每选择该项，就会按图片大小新建一个图片窗口。在非图片区域点击右键，则右键菜单中没有“导出图片”项。

为了讲解方便，本例中使用 VBScript 脚本程序直接调用可执行文件，这会造成部分杀毒程序提示发现可疑脚本。标准方法是将程序编译为 ActiveX DLL 文件，将其注册至系统中，通过调用该对象实现功能，这样就不会提示发现可疑脚本了。

**注意：**脚本文件没有对覆盖热点的图片或框架页做判断，因此无法正确读取被热点覆盖的图片文件或隐藏框架页图片。

“HTTP 读文件 ()”命令除了能读取可下载的字节集文件外，还可以访问 ASP、PHP 等动态网页，返回处理后的 HTML 超文本源代码。返回的结果就是在浏览器中所查看的网页源代码。现在就来看一个利用“HTTP 读文件 ()”命令读取网页 HTML 源代码实现网络数据采集的例子。





本实例借助 TOM.com 的 mp3 搜索引擎，用程序自动收集整理搜索到的音乐地址，实现在线音乐地址搜索功能。实现效果如图 9-11 所示。



图 9-11 在线音乐收集器

## 第一步：分析网站

首先来分析 TOM.com 的网页访问地址格式

TOM.com 的 mp3 搜索引擎首页地址为：<http://search.tom.com/mp3/>，在首页填入歌曲或歌手名称，比如“阿杜”，点击“搜索”按钮，则自动打开一个新窗口，显示搜索结果第一页内容。如图 9-12 所示。

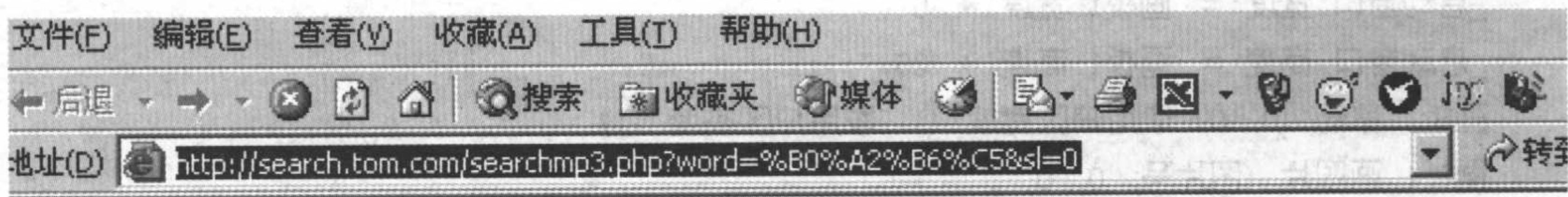


图 9-12 搜索窗口地址栏

word=%B0%A2%B6%C5 代表其搜索关键词为%B0%A2%B6%C5（汉字“阿杜”的 URL 译码）。如果想看第二页结果，可以点击页面下方链接跳转。鼠标移到各跳转页面链接上可以看到如表 9-3 所示链接地址。

第一页地址	<a href="http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=1&amp;tomsearch=mp3">http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=1&amp;tomsearch=mp3</a>
第二页地址	<a href="http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=2&amp;tomsearch=mp3">http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=2&amp;tomsearch=mp3</a>
第三页地址	<a href="http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=3&amp;tomsearch=mp3">http://search.tom.com/searchmp3.php?sl=0&amp;word=阿杜&amp;class=&amp;bg=3&amp;tomsearch=mp3</a>

表 9-3 各搜索页超链接地址

从中可以看出，只要每次变换 bg 的值，就可以取得不同搜索页面的结果了。

接下来就是如何判断搜索结果的最后一页了，可以把 bg 的值设为 100 试试，结果如图 9-13 所示。



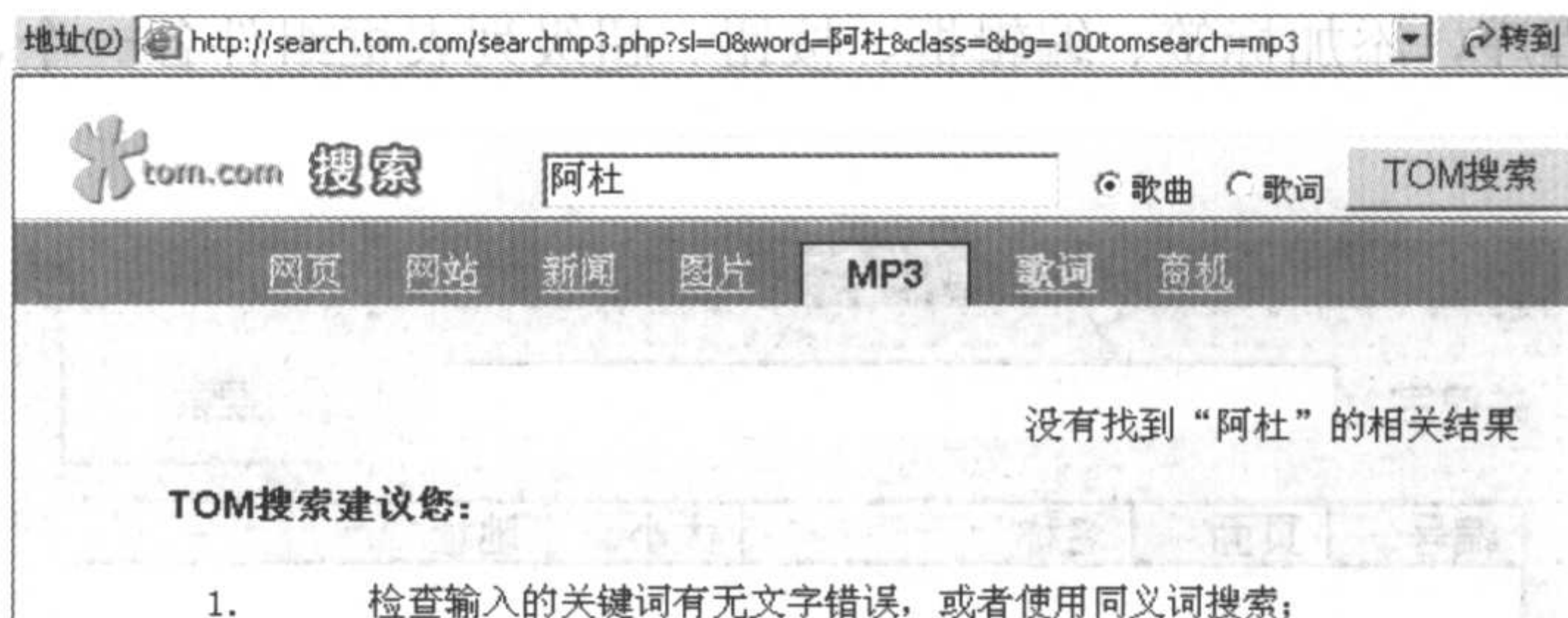


图 9-13 返回错误页面

可以看到, 网站提示“没有找到…的相关结果”。只需要不断增加 bg 参数值, 直到返回的页面中包含“没有找到…的相关结果”文本内容, 即可认为搜索完毕。

根据 HTTP 协议规定, 如果 URL 参数中存在“空格”符号, 一律用“+”号代替。如图 9-14 所示。

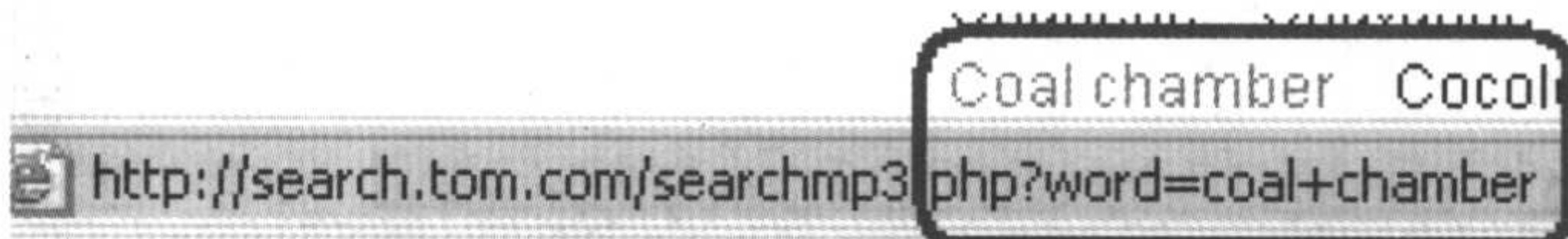


图 9-14 参数中空格一律用“+”号代替

接下来, 分析每个页面的代码内容。在 IE 菜单选择“查看”→“源文件”, 可以看到如图 9-15 所示的 HTML 代码。

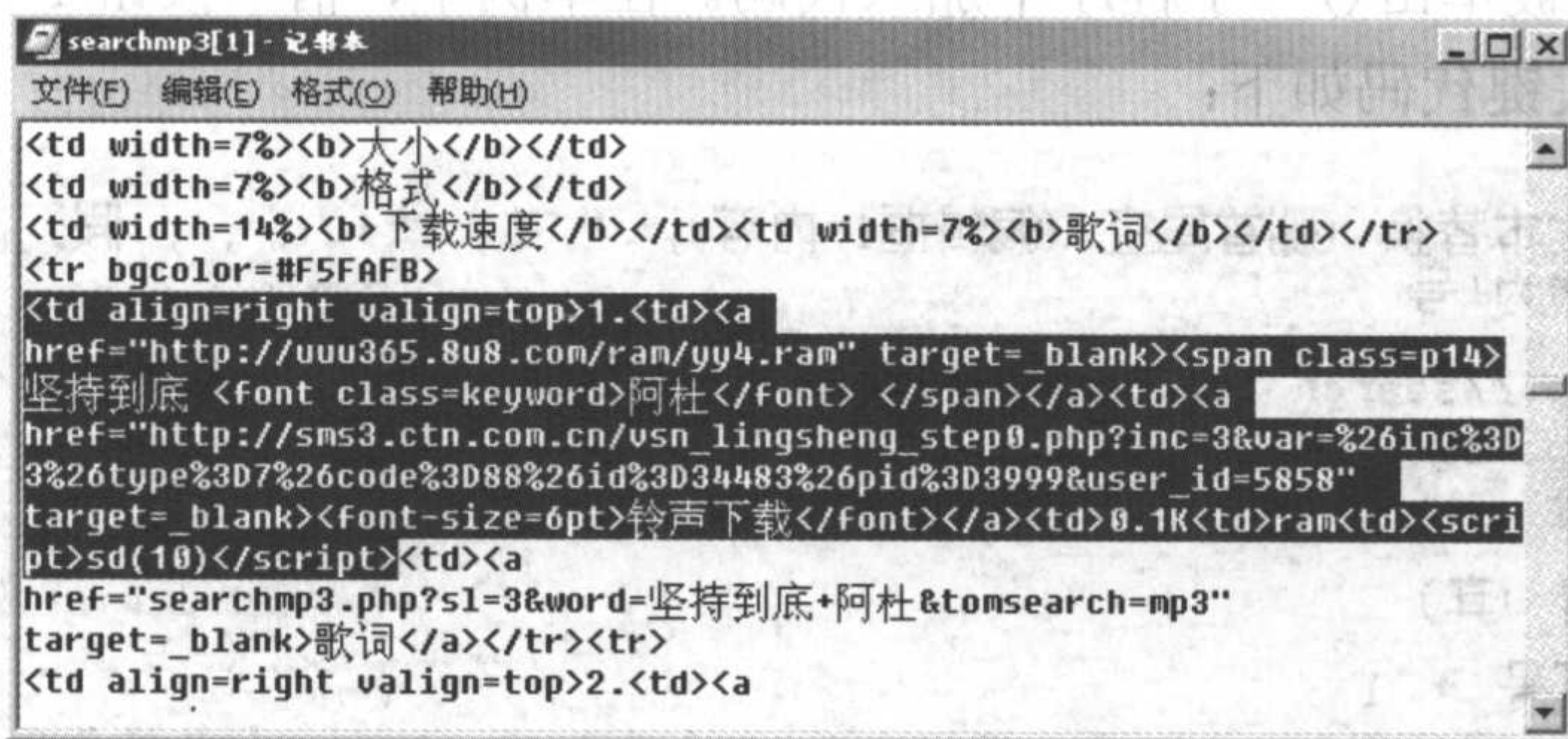


图 9-15 页面源代码

ASP 或 PHP 页面脚本开发和易语言一样, 显示重复格式数据时会采用循环语句自动列表显示, 因此, 每条记录的 HTML 代码格式都几乎一模一样。只要分析清楚一条记录的格式, 就可以用循环语句自动获取其他记录的内容。反选部分代码是分析后所得到的一条记录的代码段。表现形式如图 9-16 所示。

歌曲名 (点击可下载或播放音乐)	铃声	大小	格式	下载速度	歌词
1. 撕夜 阿杜	铃声下载	4.4M	mp3	——	歌词

图 9-16 网页表现形式

网站分析完毕, 现在开始着手编写程序。





新建一个易程序，添加标签、编辑框、按钮、超级列表框组件各一个，按图 9-17 设计界面。

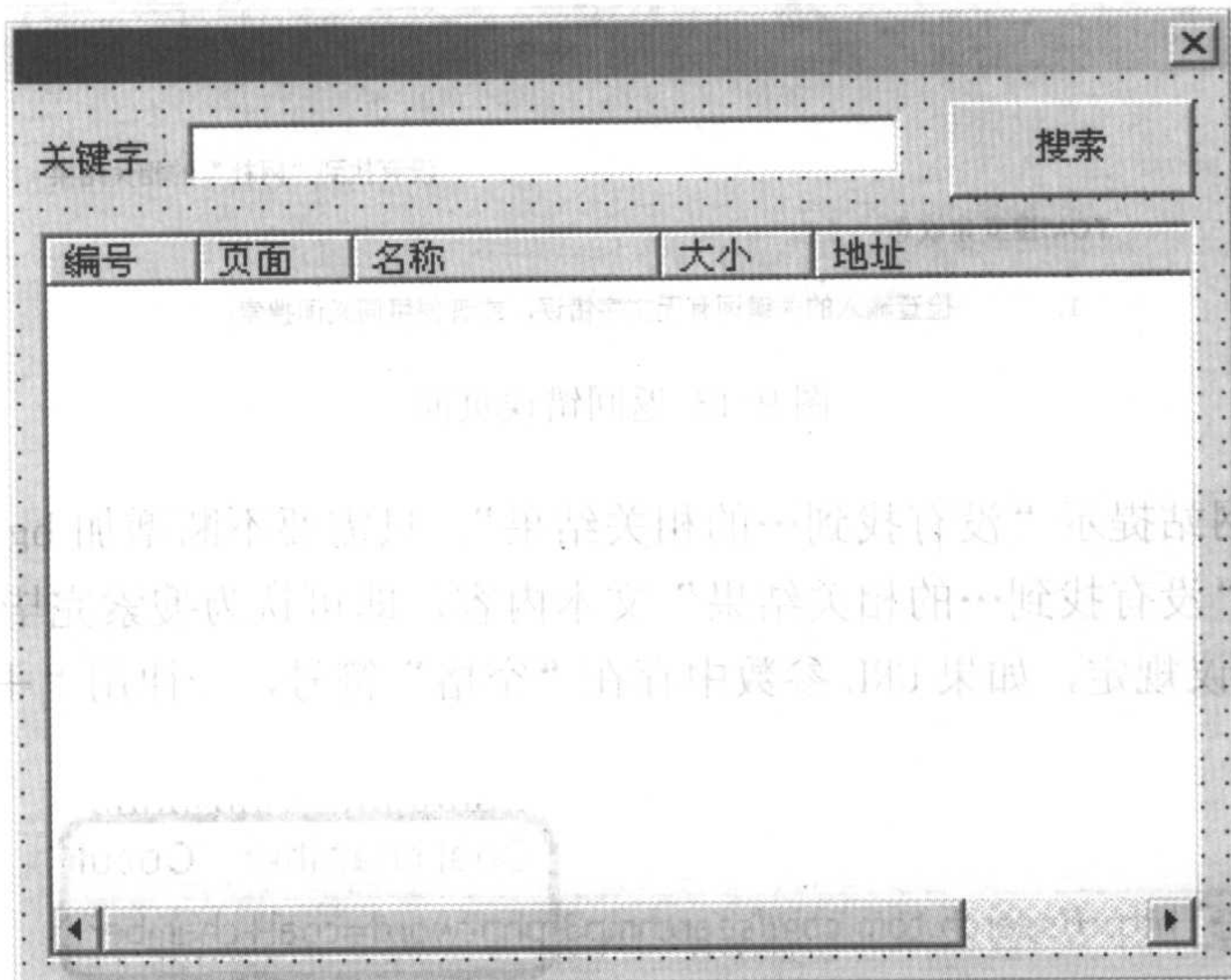


图 9-17 音乐搜索器设计界面

在程序中增加一个整数型程序集变量“编号”作为记录的顺序流水号。

在“\_按钮 1\_被单击()”子程序中加入代码。在本例中，借“按钮 1”的标题属性作为停止判断标志。关键代码如下：

```
关键字 = 子文本替换 (删首尾空 (编辑框1.内容), " ", "+", , , 假)
' 将空格符号转为+号

地址 = "http://search.tom.com/searchmp3.php?sl=0&word=" + 关键字 + "&class=&to
msearch=mp3&bg="
' 建立初始地址

--> 判断循环首 (真)
    页号 = 页号 + 1
    页面内容 = 从字节集转换 (HTTP读文件 (地址 + 到文本 (页号)), #文本型)
    --- 如果真 (寻找文本 (页面内容, "没有找到", 1, 真) > 0)
        跳出循环 ()
    页面处理 (页面内容, 页号)
    --- 如果真 (按钮1.标题 = "搜索")
        跳出循环 ()

--- 判断循环尾 ()
--- 信息框 ("搜索完成", 0, )
```

新建一个“页面处理”用户自定义子程序，此子程序实现对每个页面进行分析处理。在处理代码前，尽量减少页面内容，排除干扰，如本例分析页面子程序开始处理部分，就



跳过了页面源代码的头部和尾部，尽量只保留重复格式的记录段，提高处理准确度。

```
内容 = 取文本左边 (内容, 寻找文本 (内容, "<!-- 搜索结果 结束 -->", 1, 真))
'分析源代码，截去不必要的代码，以免影响程序判断
尾位置 = 寻找文本 (内容, "<b>歌曲名</b>", , 真)
'从页面中部开始搜索
```

因为页面处理代码是在分析页面源代码基础上得出的，因此查找每一条记录的首尾文本位置时，必须严格按照网页源代码的结构进行判断。在选取几个标志性文本时，一定要对照源代码前后细心考虑，否则很容易得到错误的结果。

```
首位置 = 寻找文本 (内容, "<a href=" + #引号 + "http://", 尾位置, 真)
'搜索歌曲超链接起始位置
--- 如果 (首位置 > 0)
    尾位置 = 寻找文本 (内容, #引号, 首位置 + 9, 真)
    '搜索歌曲超链接结束位置
    文本容器 = 取文本中间 (内容, 首位置 + 9, 尾位置 - 首位置 - 9)
    '取出歌曲超链接文本
    如果真 (寻找文本 (文本容器, "http://sms3.ctn.com.cn", 1, 真) < 0)
        '除去铃声下载超链接地址的影响
        编号 = 编号 + 1
        索引值 = 超级列表框1.插入表项 (, 到文本 (编号), , , )
        超级列表框1.置标题 (索引值, 1, 到文本 (页面号))
        超级列表框1.置标题 (索引值, 3, 文本容器)
        尾位置 = 寻找文本 (内容, "</span>", 尾位置, 真)
        '搜索歌曲名称结束位置
        首位置 = 倒找文本 (内容, "<span class=p14>", 尾位置, 真)
        文本容器 = 取文本中间 (内容, 首位置 + 16, 尾位置 - 首位置 - 16)
        '取出歌曲名称
        文本容器 = 子文本替换 (文本容器, "<font class=keyword>", , , , 假)
        文本容器 = 子文本替换 (文本容器, "</font>", , , , 假)
        '过滤原始代码
        超级列表框1.置标题 (索引值, 2, 文本容器)
```

具体代码参见随书光盘中“在线音乐搜索器.e”。

通过以上几个实例，可以看出，要想做好网络应用型程序，除了要对易语言自带命令函数了解透彻，还要尽可能多的了解协议相关的知识。网络的资源是无尽的，只要能多些思路，就可以通过有限的程序命令，开发出功能丰富的软件。

### 9.3 数据通讯程序

本章把基于自定义数据通讯协议的程序，如聊天室、网络监控、网络游戏等应用类程序统称为数据通讯型程序。数据通讯是网络应用的基础，任何网络程序都需要通过网络来





交换数据。虽然为了实现网络无界限而制订了很多通讯协议标准，但在实际应用中，基于简化编程、通讯速度和数据安全的需要，仍有可能必须自行定义网络数据交换的法则。

在这里，仅就基于 TCP/IP 和 UDP 协议进行数据通讯的程序进行研讨，基于其他底层通讯协议的相关内容请自行查阅资料。

### 9.3.1 网络通信命令

开发数据通讯型程序常常要获取自己或对方的网络地址 (IP)，比如通过 IP 连接进行网络对战游戏，或通过 IP 连接登录聊天室等，都要知道主机方 (建立游戏或聊天室的一方) 的 IP 才能连接。

主机名可用做在局域网通讯中标识本机。

“取主机名 ()” 返回本机的主机名。使用格式如下：

文本变量 = 取主机名 ()

“通信测试 ()” 测试与指定主机是否能够正常通讯。返回被测试主机的通讯响应时间。如果无法通讯或者测试失败，返回 -1。实例代码如下：

文本变量 = 到文本 (通信测试 (IP 编辑框 1. 地址, ))

“转换为主机名 ()” 将指定的 IP 地址转换为其主机名。如果失败返回空文本。实例代码如下：

文本变量 = 转换为主机名 (IP 编辑框 1. 地址)

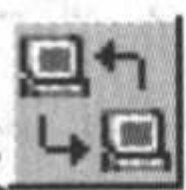
“转换为 IP 地址 ()” 将指定的主机名转换为其 IP 地址。如果失败返回空文本。实例代码如下：

文本变量 = 转换为 IP 地址 (主机名变量)

以上是网络数据通讯最基本的命令。

### 9.3.2 数据通讯组件

数据报



数据报是一种小数据量的网络数据交互方式，发送数据之前无需和对方建立连接，因此要比使用客户/服务器控件传输数据要快，而且它能够以广播方式发送信息，非常适合局域网中互传消息之用。但由于是非连接方式，因此只能判断是否发送成功，无法判断是否接收成功。如果一次性发送的数据量过大，有可能会发送失败。易语言支持一次性 1024 个字节的发送，但在实际应用中，最好不要超过 127 个字节。

如需要进行大数据量传送或可靠方式传送，请使用客户/服务器组件。

#### 1. 数据报重要属性

“端口”属性，是指定监听数据到达的端口号，可以是大于 0 小于 32767 的任何自定数值，默认值是 19730。

计算机中总共有  $256 \times 256$  即 65536 个端口 (编号从 0 到 65535)，其中前 1024 个端口都有确切的定义，对应着系统相关的一些服务。1024 号以后的端口是为应用程序保留的，可以随意选用 (易语言中限 32767 以下)。在为数据报组件 (包括客户/服务器组件) 指定



端口属性时，尽量选取较大的端口号，以免和其他软件冲突。

## 2. 数据报的专有方法

### “发送数据 ()”方法

功能：发送数据到指定主机的指定端口。

语法：数据报名称. 发送数据([接收主机地址], 接收主机端口号, 欲发送数据)

参数 1：接收主机地址——文本型，可以为主机名、IP 地址等。如果省略本参数或者提供空文本，则在指定端口广播欲发送数据。

参数 2：接收主机端口号——整数型，必须是对方（接收主机）数据报组件的端口属性指定的数值。

参数 3：欲发送数据——欲发送的数据，可以是文本型、整数型、小数型、逻辑型、日期时间型等（数据类型不限）。

返回值：逻辑型。如果数据发送成功，返回“真”；如果发送失败，返回“假”。

“发送数据 ()”方法的应用实例如下：

数据报 1. 发送数据 (“127.0.0.1”, 19730, 123)

数据报 1. 发送数据 (“mypc”, 19730, “语言”)

第一行代码表示为用 19730 端口向 IP 地址为 127.0.0.1 的主机发送整数型数据 123；

第二行代码表示为用 19730 端口向主机名为“mypc”的主机发送文本数据“语言”；

**注意：**本方法的逻辑型返回值只是表明“发送”数据的成功与否，与数据能不能成功“到达”没有任何关系。

### “取回数据 ()”方法

功能：取回数据报组件所接收到的数据。

语法：数据报名称. 取回数据()

返回值：字节集型。注意是字节集型，使用时有可能需要进行数据类型转换。

取回数据方法的应用实例如下：

数据=数据报 1. 取回数据 () //取回数据并存入字节集型变量“数据”中

编辑框 1. 内容=从字节集转换(数据, #文本型) //显示到编辑框 1 中

“从字节集转换(数据, #文本型)”表示将字节集型变量“数据”转换成文本型（第二个参数“#文本型”指定目标数据类型，“#文本型”是易语言系统定义的常量）。

上述实例也可写到一行中：

编辑框 1. 内容=从字节集转换(数据报 1. 取回数据 (), #文本型)

**注意：**字节集是比较特殊的数据类型，易语言专门提供了针对它的类型转换命令：“从字节集转换 ()”和“转换为字节集 ()”，并且还有很多专门针对字节集型数据进行操作的命令，可以查看易语言支持库面板中的字节集操作部分。“取回数据 ()”方法只能在数据报组件的“数据到达”事件的处理子程序中调用。在其他地方即使调用也无法取到正确的数据。

## 3. 数据报的重要事件

### “数据到达”事件

事件的产生时机：当有数据到达时自动产生此事件。





在本事件的处理子程序中，一个最重要的任务就是取回“到达的数据”——调用“取回数据()”方法。

## 4. 数据报的应用

通过前面的学习，初步了解了“数据报”组件每个方法和事件的作用。大家打开书中例程“数据报.e”。如图 9-18 所示。

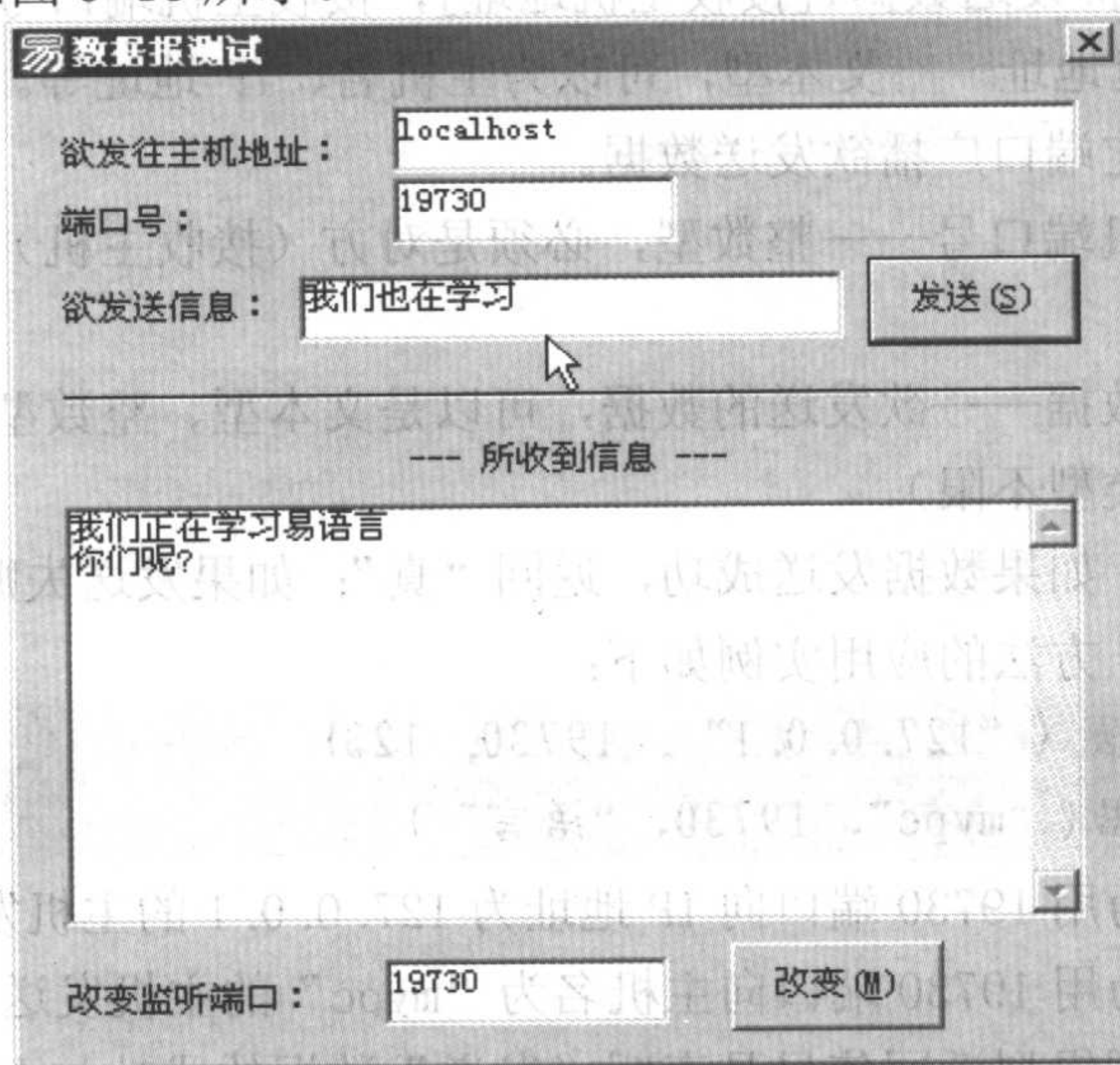


图 9-18 “数据报.e”程序界面

运行程序，填写欲发送信息文本，点击“发送”按钮，可以看到所收到的信息处就会把收到的内容显示出来。

在“\_发送按钮\_被单击”事件子程序中，使用“数据报 1. 发送数据()”方法，把信息内容发送到主机地址栏指定的计算机。程序代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
端口号	整数型			

端口号 = 到数值 (端口号编辑框. 内容)

如果真 (端口号 ≤ 0)

信息框 (“请输入欲发送数据到端口号!”, 0, )

返回 0

如果真 (数据报 1. 发送数据 (主机地址编辑框. 内容, 端口号, 发送编辑框. 内容) = (假))

信息框 (“发送失败!”, 0, )

发送编辑框. 内容 = “”

发送编辑框. 获取焦点 ()

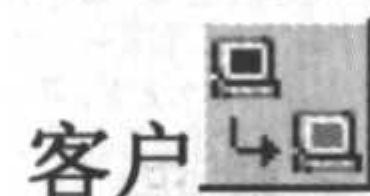
当对方的数据报接收到数据时，“\_数据报 1\_数据到达”事件子程序被触发。“数据报



1. 取回数据 ( )”取得接收到的数据，由于通过网络传输的数据都是字节集型，必须通过转换，显示到编辑框中。程序代码如下：

子程序名	返回值类型	公开	备注
_数据报1_数据到达			

收到信息编辑框.加入文本 (从字节集转换 (数据报1.取回数据 ( ), #文本型) + #换行符)



客户组件和服务端组件是一对好搭档，它们总是成对使用，一般在两个独立的应用程序中分别调用，一个是客户端程序，一个是服务器端程序，客户端应用程序主动与服务器端应用程序进行连接。(当然也可以在一个应用程序中既使用客户组件，又使用服务器组件，这样它即充当客户端又充当服务器端。)

### 1. 客户组件的重要属性

客户组件只有“名称”、“备注”、“左边”、“顶边”、“宽度”、“高度”与“标记”基本属性，没有其他的属性。它也是一种运行时不可见的组件，所以这些属性都可以不用设置。

### 2. 客户的专有方法

#### “连接 ( )”方法

功能：连接到指定主机的指定端口。

语法：客户组件名.连接(服务器地址，服务器端口号)

参数 1：服务器地址——文本型，可以为主机名、IP 地址；

参数 2：服务器端口号——整数型，即服务器组件的端口属性指定的值。

很显然，“客户”组件必须确切地知道对方的服务器地址和端口号才有可能连接到对方。一般来说，客户端程序和服务器端程序都是由同一个人（或团体）开发和部署的，服务器地址和端口号都是很容易知道的。另外，如果要在广域网（区别于局域网）范围中通信，服务器端程序所在电脑必须具有“外部 IP 地址”。

“连接 ( )”方法的应用实例如下：

客户 1.连接 (“110.0.0.1” , 19730)

与 IP 地址为 110.0.0.1 的主机中的服务器组件进行连接。该服务器组件的端口属性必须是 19730，且它所在的程序正在运行，否则连接不可能成功。

**注意：**同一时刻，一个客户组件只能连接一个服务器组件；一个服务器组件可以同时跟多个客户组件连接。

#### “断开连接 ( )”方法

功能：断开与服务器的连接。

语法：客户名称.断开连接 ( )

“断开连接 ( )”方法的应用实例如下：

客户 1.断开连接 ( )





### “发送数据( )”方法

功能：向已经建立连接的服务器组件发送数据。

语法：客户组件名.发送数据(数据)

参数可以是各种类型的数据。

“发送数据”方法的应用实例如下：

客户 1.发送数据(0)

客户 1.发送数据(编辑框 1.内容)

客户 1.发送数据(图片框 1.图片) //分别向服务器发送整数、文本、字节集数据。

**注意：**必须先跟服务器组件正确建立连接之后，才能发送数据。

### “取回数据( )”方法

功能：取回服务器发送来的数据。

语法：客户组件名.取回数据( )

本方法返回字节集型的数据。使用时有可能需要进行数据类型转换（用“从字节集转换( )”命令），这跟数据报组件的“取回数据( )”方法是完全一致的。

## 3. 客户组件的重要事件

### “数据到达”事件

当服务器端将数据发送过来后产生本事件。在本事件的处理子程序中调用“取回数据( )”方法即可取回本次所收到的数据。

### “连接断开”事件

当连接被服务器端断开后会产生本事件。连接断开后，不能继续发送数据，除非重新建立连接。

## 4. 客户与服务器的交互

(1) 首先由客户组件建立与服务器组件的连接（通过调用“连接( )”方法），这时服务器组件产生“客户进入”事件，并在该事件中获得客户的地址（通过调用“取回客户( )”方法）。此后双方可以互相发送、接收数据。

(2) 客户和服务器组件都可以在连接断开之前的任何时间调用“发送数据( )”方法向对方发送数据，此时对方产生“数据到达”事件。在“数据到达”事件中调用“取回数据( )”方法即可取得对方发送来的数据。

(3) 客户组件可调用“断开连接( )”方法断开与服务器组件的连接，此时服务器组件产生“客户离开”事件。服务器组件可调用“断开客户( )”方法断开与客户组件的连接，此时客户组件产生“连接断开”事件。

## 服务器

### 1. 服务器组件的重要属性

#### “端口”属性

整数型，指定监听数据到达的端口号，可以是大于 0 小于 32767 的任何自定义数值（应尽量取大于 1024 的较大值）。其含义与数据报组件的同名属性类似。



服务器与客户的端口号必须一致，否则不能相互通讯。

## 2. 服务器组件的专有方法

### (1) “取回客户( )”方法

功能：取回与服务器连接的客户地址。

语法：服务器名称. 取回客户( )

本方法返回一个文本型的值，其中记录了客户的地址（IP 地址 + 端口）。当服务器组件向客户发送数据或断开客户时，都需要指定该地址。

“取回客户( )”方法的应用实例如下：

```
客户地址=服务器 1. 取回客户( )
```

取回客户的地址，并保存到文本型变量“客户地址”中。“客户地址”要事先定义为文本型全局变量或程序集变量，以供其他子程序使用。

**注意：**通常在服务器组件的“客户进入”或“客户离开”（特别是“客户进入”）事件的处理子程序中调用本方法。

### (2) “取回数据( )”方法

功能：取回客户发送来的数据。

语法：服务器名称. 取回数据( )

返回值为字节集型。

### (3) “发送数据( )”方法

功能：向指定客户发送数据。

语法：服务器名称. 发送数据(接收客户，数据，[最长等待时间])

参数 1：接收客户——文本型，必须是调用“取回客户( )”方法获得的客户地址。

参数 2：数据——可以是各种类型的数据。

参数 3：最长等待时间——指定等待发送成功的最长时间，单位为秒。如果省略本参数，默认为无限等待。

“发送数据( )”应用实例如下：

```
服务器 1. 发送数据(客户地址, 123, )
```

向客户端发送整数型数据 123。这里的“客户地址”就是前面调用“取回客户( )”时的返回值。

### (4) “断开客户( )”方法

功能：与指定客户断开连接。

语法：服务器名称. 断开客户(欲断开客户)

参数必须是调用“取回客户( )”方法所返回的客户地址文本。

“断开客户( )”应用实例如下：

```
服务器 1. 断开客户(客户地址)
```

与“客户地址”所指定的客户断开连接。这里的“客户地址”就是前面调用“取回客户( )”方法的返回值。

## 3. 服务器的重要事件





### (1) “数据到达”事件

当服务器端将数据发送过来后产生本事件。在本事件的处理子程序中调用“取回数据( )”方法即可取回本次所收到的数据。

### (2) “客户进入”事件

当有新客户连接当前服务器组件后产生本事件。本事件处理子程序的一个重要的任务就是：调用“取回客户( )”方法获得新客户的地址，并保存到文本型的全局变量或程序集变量中，供以后使用（服务器组件的方法“发送数据( )”、“断开客户( )”都需要指定客户的地址）。

### (3) “客户离开”事件

当客户端与服务器端的连接断开后，会产生本事件。在本事件的处理子程序中调用“取回客户( )”方法即可取回此客户的地址（IP 地址 + 端口）。

## 4. 客户与服务器组件应用

同时运行随书光盘中的“聊天室服务器.e”和多个“聊天室客户端.e”例程。程序运行效果如图 9-19 和图 9-20 所示。

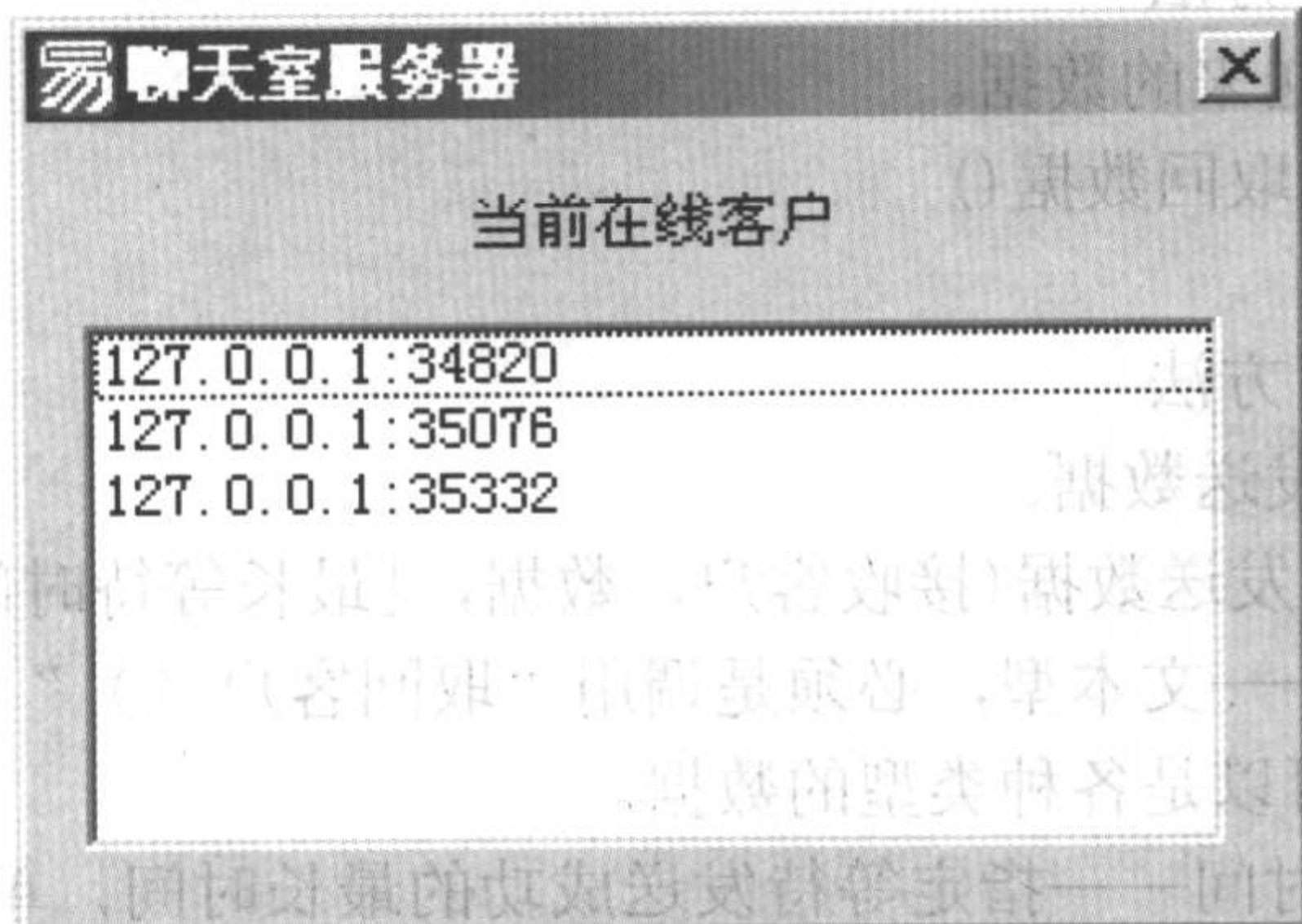


图 9-19 服务器端

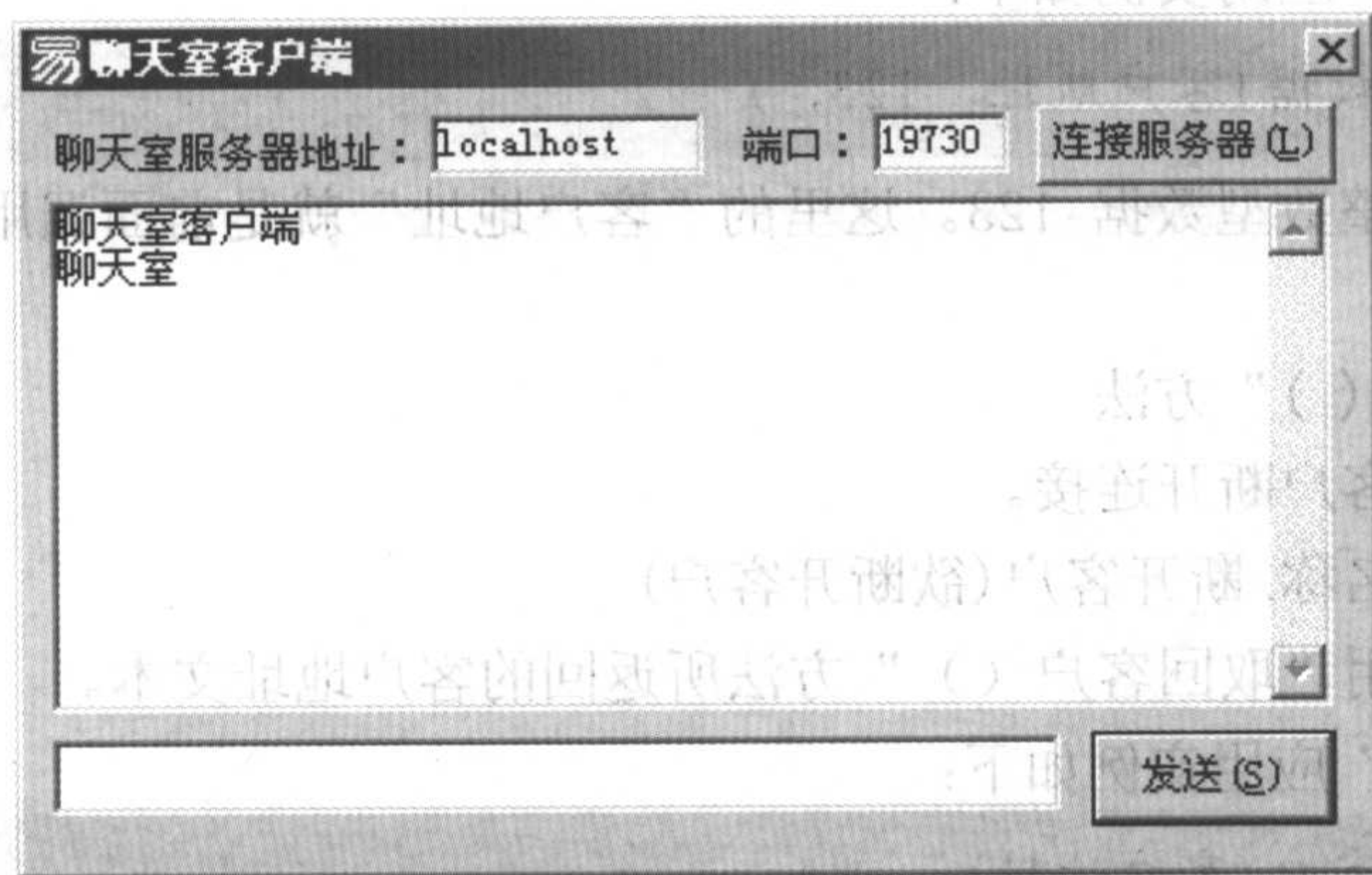


图 9-20 其中一个客户端

只要客户端连接，服务器端就会取得客户端的地址并显示在列表中，相关程序代码如下：



子程序名	返回值类型	公开	备注
_服务器1_客户进入			

列表框1.加入项目 (服务器1.取回客户 (), )

客户连接到服务器后，把数据发送到服务器端，然后用服务器把数据发送到所有客户端。相关程序代码如下：

子程序名	返回值类型	公开	备注
_服务器1_数据到达			

变量名	类型	静态	数组	备注
项目数	整数型			
计次变量	整数型			
收到数据	字节集			

将所收到的信息转发给所有在线客户。

收到数据 = 服务器1.取回数据 ()

项目数 = 列表框1.取项目数 ()

---▶ 计次循环首 (项目数, 计次变量)

服务器1.发送数据 (列表框1.取项目文本 (计次变量 - 1), 收到数据, 1)

--- 计次循环尾 ()

按客户端的数量和客户端地址依次发送取回的数据。这样，每个客户端都可以收到相同的信息并显示在信息栏中。如果需要加入身份判断，可以在客户端连接服务器时，由服务器返回其客户信息，这样就可以使用客户信息作为身份的识别标志了。

### 9.3.3 网络通讯支持库

网络通讯支持库是为了增强原有网络通讯组件（命令）的功能而提供的，其基本操作和使用方法与原网络通讯组件（命令）类似，互为补充。在实际应用中服务器组件适合搭建服务平台，而网络服务器数据类型则适合一对一的大文件传输，两者可互相搭配使用。

#### 网络通讯支持库命令

“取本机名 ()”与原网络通信命令“取主机名”功能相同。使用格式如下：

文本变量 = 取本机名 ()

“取本机 IP ()”返回本机所有绑定的 IP 地址列表。实例代码如下：

IP 列表 = 取本机 IP () ' IP 列表为文本型动态数组

#### 网络通讯支持库数据类型

网络数据报、网络服务器和网络客户端这三个数据类型可以简单地理解为无界面的网络通讯组件。相对于原有的网络通讯组件来说，这三个数据类型更适合一对一数据传输的环境。大家直接来看两个例程，深入了解这三个数据类型的使用方法。

打开随书光盘本章目录中的“网络数据报.e”，按 F5 运行，运行界面如图 9-21 所示。



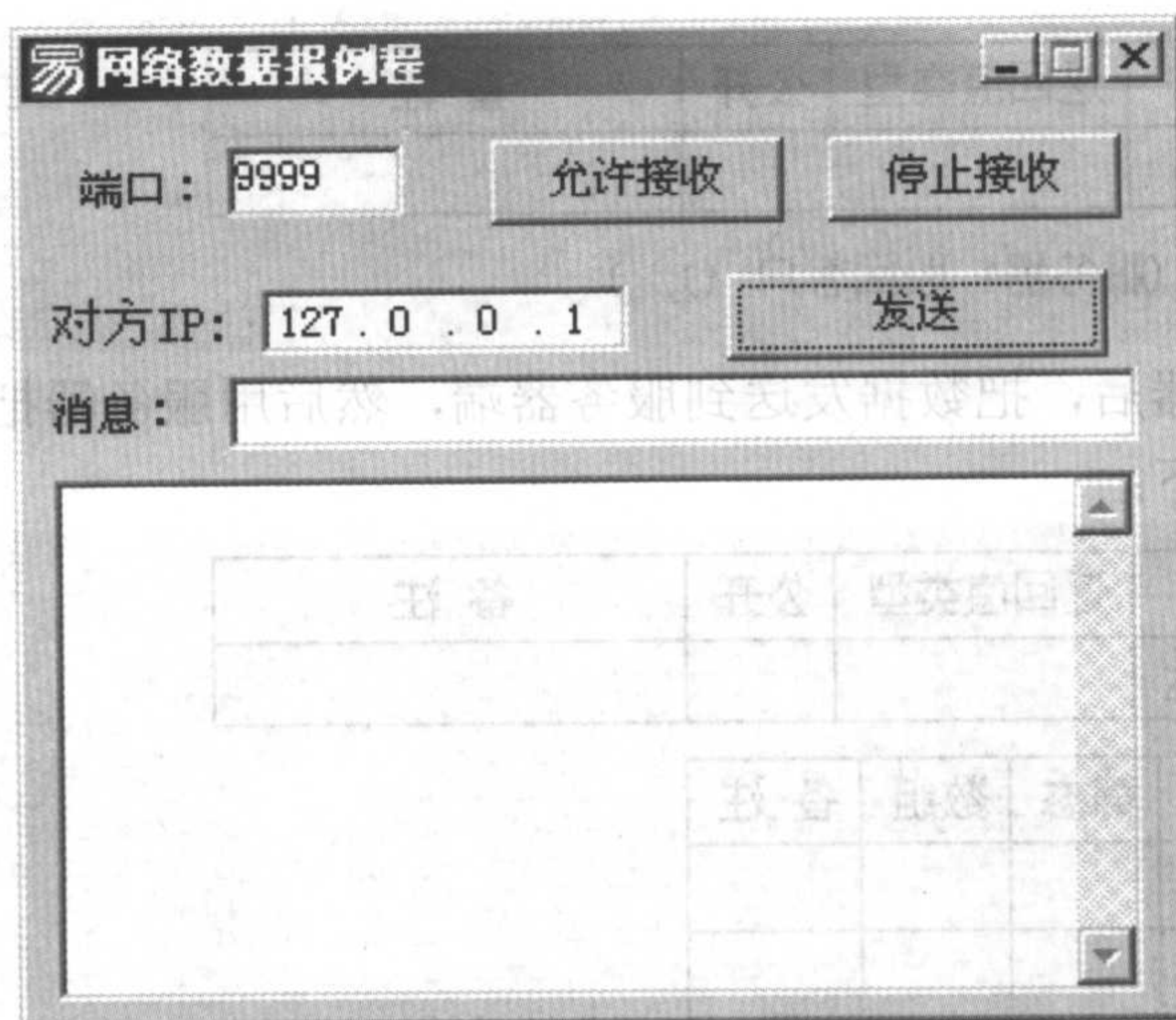


图 9-21 “网络数据报.e”运行界面

下面分析一下几个功能按钮的代码。

子程序名	返回值类型	公开	备注
_允许接收_被单击			

```
--- 如果真 (到数值 (端口框.内容) = 0)
    信息框 (“端口号设置不正确”, 0, )
    返回 0
端口框.禁止 = 真
--- 如果真 (数据报.配置 (到数值 (端口框.内容)))
    _启动窗口.标题 = “正在监听”
    监听标志 = 真
    启动线程 (@子程序1)
```

在“允许接收”按钮被单击事件中，使用“配置”方法来启动网络数据报的监听服务，在监听期间，不允许修改端口号。启动成功后，创建了一个子程序1的线程。

子程序名	返回值类型	公开	备注
子程序1			

变量名	类型	静态	数组	备注
字节集容器	字节集			
接收方信息	对方信息			

```
--- 判断循环首 (监听标志)
    字节集容器 = 数据报.接收 (15, 接收方信息)
    每15秒检测一次监听服务标志, 避免无限等待
    --- 如果真 (字节集容器 ≠ 取空白字节集 (0))
        消息框.加入文本 (接收方信息.对方IP + “:” + 到文本 (字节集容器) + #换行符)
        IP编辑框1.地址 = 接收方信息.对方IP
    --- 判断循环尾 0
```



上述程序中的“对方信息”类型是网络通讯支持库新定义的数据类型。大家可以查看支持库面板中的此支持库中的数据类型定义。

在子程序 1 中不断的循环准备接收消息，每 15 秒检查一次服务停止标志。在此期间，如果接收到信息，就加入到消息框中，并自动设置对方 IP 为回复 IP。

子程序名	返回值类型	公开	备注
_停止接收_被单击			

```

--- 如果真 (数据报.关闭 ()
    _启动窗口.标题 = “停止监听”
    监听标志 = 假
    端口框.禁止 = 假
    
```

当点击“停止接收”按钮后，将监听标志设为“假”，并允许修改端口号。

子程序名	返回值类型	公开	备注
_发送_被单击			

变量名	类型	静态	数组	备注
对方信息	对方信息			

```

--- 如果真 (到数值 (端口框.内容) = 0)
    信息框 (“对方端口不正确”, 0, )
    返回 0
    对方信息.对方IP = IP编辑框1.地址
    对方信息.对方端口 = 到数值 (端口框.内容)
    --- 如果 (数据报.发送 (对方信息, 取本机名 () + “:” + 发送框.内容, 15) = 假)
        信息框 (“发送失败”, 0, )
        发送框.内容 = “”
    发送框.获取焦点 ()
    
```

发送信息超时值设为 15 秒，此处为简化讲解，删减了对方 IP 地址有效性判断代码。

通过上例可以看出，网络数据报数据类型不具备网络广播功能，因此尚无法完全替代数据报组件的功能，但其大数据量传输性能相比数据报组件更为出色，可根据需要灵活选用。

再来看看网络服务器和网络客户端的使用实例。

打开随书光盘“网络服务器.e”和“网络客户端.e”，分别按 F5 运行，运行界面如图 9-22、图 9-23 所示。

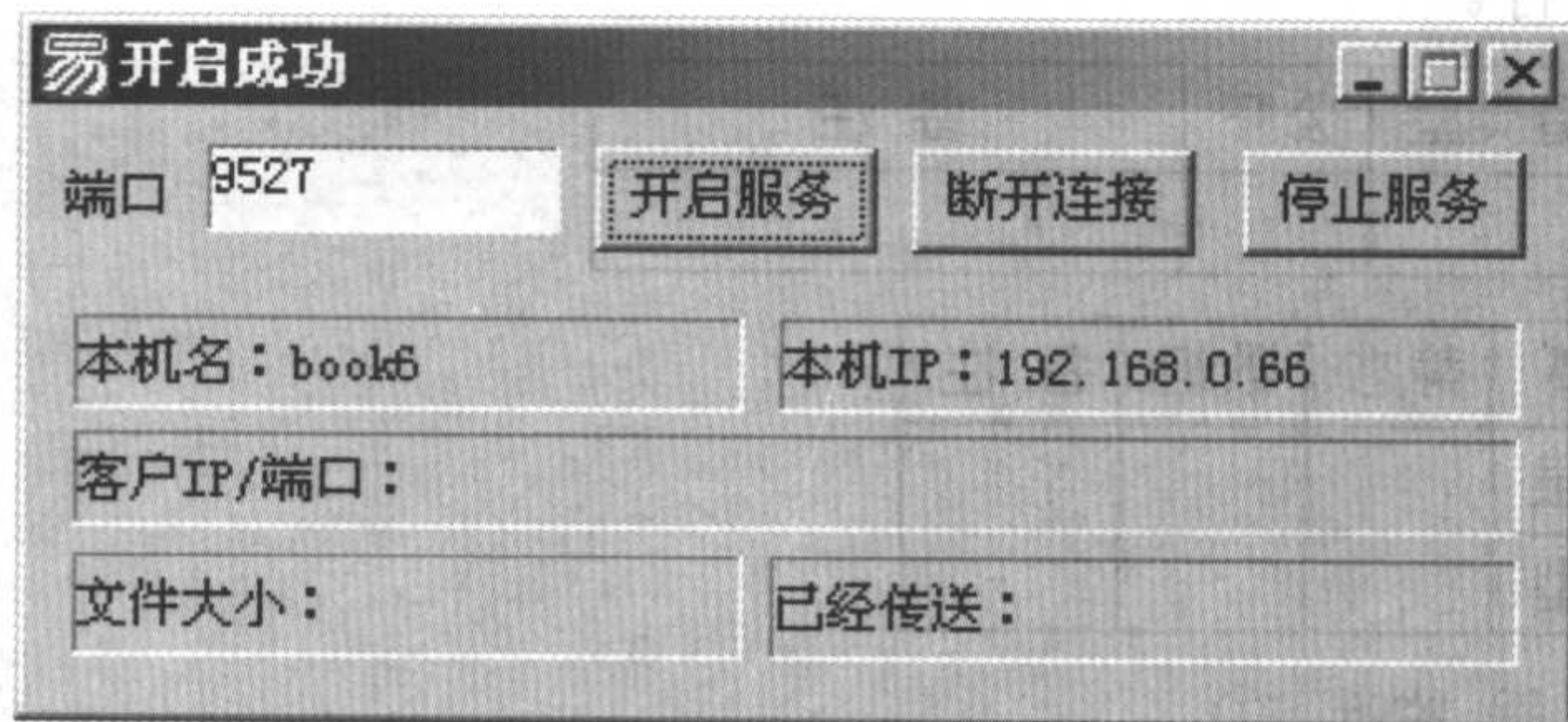


图 9-22 “网络服务器.e”运行界面



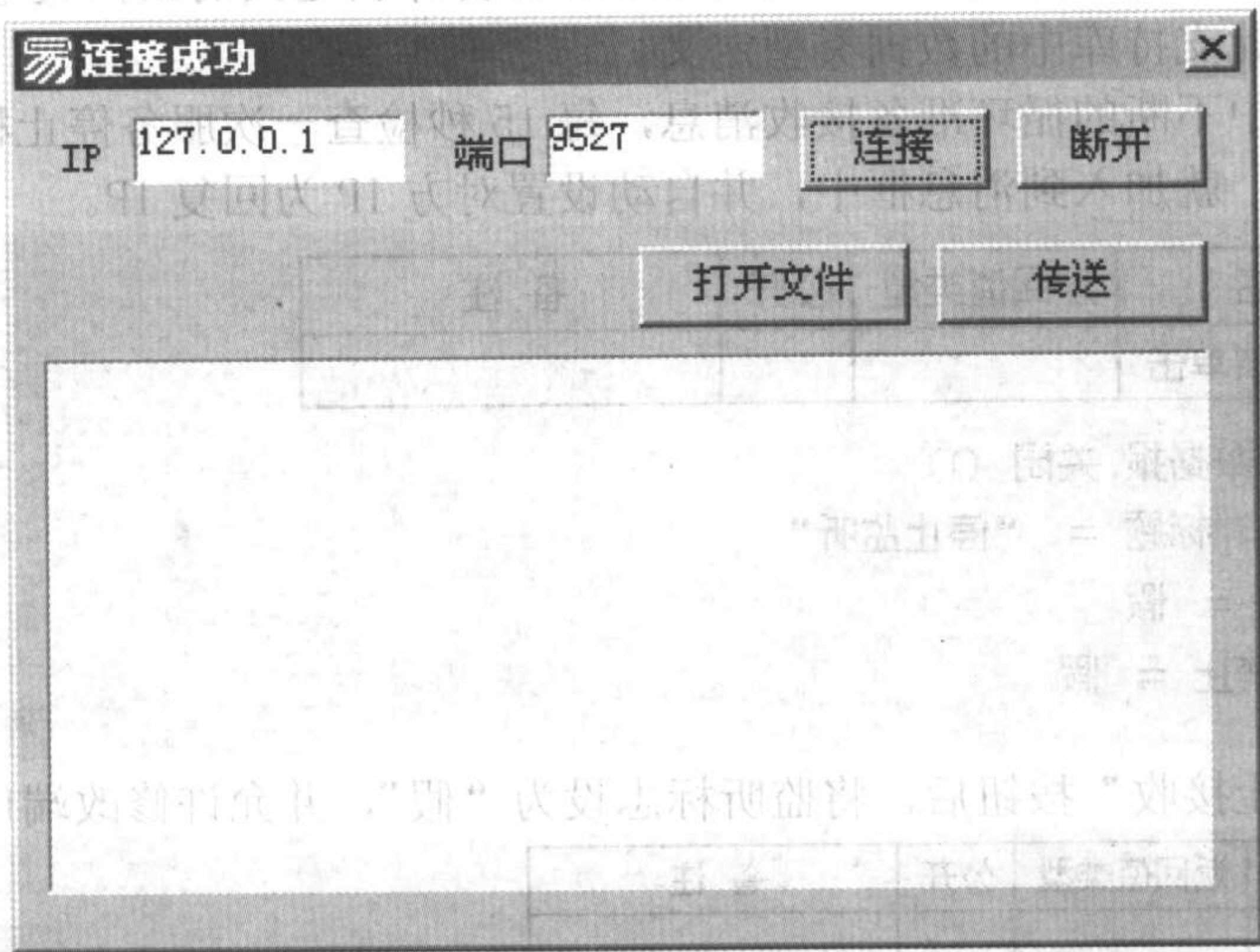


图 9-23 “网络客户端.e”运行界面

本例程实现了网络文件传输功能，网络客户端连接到网络服务器后，申请传送文件，获得服务器许可后方能传输数据。服务器端会在接收文件前指定保存位置，或者拒绝客户端文件传输请求。

大家主要学习服务器端的代码。下面是服务器启动服务的代码：

子程序名	返回值类型	公开	备注
开启服务_被单击			

```
数据 = { }
文件大小 = 0
客户句柄 = 0
如果真 (服务器.启动 (到数值 (编辑框1.内容)) = 真)
    启动窗口.标题 = “开启成功”
    启动线程 (&子程序1)
```

在这里，同样为所启动的服务单独建立一个线程来接收数据，以免等待接收客户端数据时，影响主进程运行。

子程序名	返回值类型	公开	备注
子程序1			

变量名	类型	静态	数组	备注
临时数据	字节集			
数据大小	整数型			

客户句柄 = 服务器.监听 (0)



```

标签2.标题 = “客户IP/端口:” + 到文本 (服务器.取客户IP (客户句柄))
+ “:” + 到文本 (服务器.取客户端口 (客户句柄))

```

```

通用对话框1.文件名 = 到文本 (服务器.接收 (客户句柄, 0))

```

```

---如果 (通用对话框1.打开 () = 真)

```

```

    服务器.发送 (客户句柄, “ok”, 0)

```

```

    文件大小 = 从字节集转换 (服务器.接收 (客户句柄, 0), #整数型)

```

```

    标签3.标题 = “文件大小:” + 到文本 (文件大小)

```

```

    -->循环判断首 ()

```

```

        临时数据 = 服务器.接收 (客户句柄, 0)

```

```

        数据大小 = 数据大小 + 取字节集长度 (临时数据)

```

```

        标签4.标题 = “已经传送:” + 到文本 (数据大小)

```

```

        如果真 (取字节集右边 (临时数据, 8) = 到字节集 (“send_en
        d”))

```

```

            数据 = 数据 + 取字节集左边 (临时数据, 取字节集长度 (临时数
            据) - 8)

```

```

            跳出循环 ()

```

```

            数据 = 数据 + 临时数据

```

```

    ---循环判断尾 (真)

```

```

    写到文件 (通用对话框1.文件名, 数据)

```

```

    -->服务器.发送 (客户句柄, “no”, 0)

```

在子程序 1 中, 服务器端首先不停地监听端口, 直到有一个客户端连接到端口上。随后, 服务器端停止监听, 显示其 IP 地址, 并等待客户端提交文件名。如果不保存该文件, 服务器会返回拒绝消息; 如果确定保存, 则服务器发送同意消息, 并等待客户端回传文件大小, 开始接收数据, 直到发现结束标志为止。期间, 由于大文件会自动被拆分为多个数据包传输, 因此用了一个循环判断多次提取客户端数据。

由本例可以看出, 网络服务器和网络客户端在使用方法上, 与服务器组件和客户组件没有本质的不同, 不过网络服务器更适合接收一对一的文件传输环境。由于网络服务器缺乏事件触发机制, 因此, 最好为网络客户端每个连接单独创建一个线程, 以保证实时性。在实际应用中, 一定要考虑网络服务器的定时退出监听, 检测服务标志的功能实现, 否则, 有可能因为接收进程陷入无限等待状态而影响其他进程的运行。

## 9.4 硬件通信型程序

程序通过读写端口 (串口/并口), 可以做到和硬件产品进行数据通讯。这是“软件控制硬件”重要的方式之一。由于硬件产品种类繁多, 其遵循的通讯协议也多不相同, 故本节不在具体协议上作过多纠缠。本节重点介绍在易语言中通过“端口”组件和“端口访问支持库”进行端口读写, 进而达到控制硬件目的的实际应用。



## 9.4.1 硬件通信组件

### 端口组件

端口组件通过串行端口传输和接收数据，为应用程序提供串行通讯功能。端口组件依靠两类事件来获得端口通讯消息：收到信号和数据到达。由于硬件通信的特殊性，这两种事件触发应用于不同的工作环境。

“收到信号”事件一般用于工业实时监控方面，如使用光敏传感器自动进行产品计数，当光敏传感器检测到一个产品移过时，即会发出一个信号跳变，通过本事件即可监控到该事件并自动予以计数。

“数据到达”一般用在与单片机通讯方面，如计算机与调制解调器连接、计算机与专用机通讯、视频监控摄像头云台(底座)控制等等，通过所约定的通讯协议进行数据交换。

设计时，每个端口组件对应着一个串行端口。如果应用程序需要访问多个串行端口，必须使用多个端口组件。可以在 Windows “控制面板”中改变端口地址和中断地址。

端口组件有几个属性、方法和事件相对较为重要：

“端口号”属性指定监听数据到达的端口号，可以是大于 0 小于 32767 的任何自定数值。

“波特率”属性指定通信端口所使用的波特率，比较常用的波特率有：110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000。

“奇偶校验”属性指定通信时对数据是否进行奇偶校验。

“数据位数”属性所对应的是可操作数据的位数。

“停止位数”属性可以为以下值之一：1：1 位； 2：1.5 位； 3：2 位。

“启动 ()”方法，启动或重新启动对指定端口的操作。在对端口进行操作之前必须首先启动，如在端口启动后又更改了端口属性必须重新启动。成功返回真，失败返回假。

“停止 ()”方法，关闭已经启动的指定端口。

“发送数据 ()”方法，从端口发送指定的数据。成功返回真，失败返回假。

“数据到达”事件，每当通信端口接收到一个数据字节，就会产生本事件。可与“事件字符”属性配合使用。

“事件字符”属性，指定用作触发事件的字符。只使用文本的首字符，为空表示不支持事件字符。该属性能够控制“数据到达”事件的发生条件，因为在硬件通信过程中，每收到一个字节即会触发一次到达事件，有可能因为无效信息频繁影响主程序的运行，也不利于数据的分析处理。由于无法预测数据的发生时间和长度，因此可以通过“事件字符”属性指定数据开始标志和结束标志。

来看一个电话拨号程序的例子，本例程需要一个可拨打电话的调制解调器。打开随书光盘本章目录中“电话拨号.e”。

按图 9-24 所示在窗体中放入按钮组件四个、标签组件、编辑框组件、组合框组件及端口组件各一个。



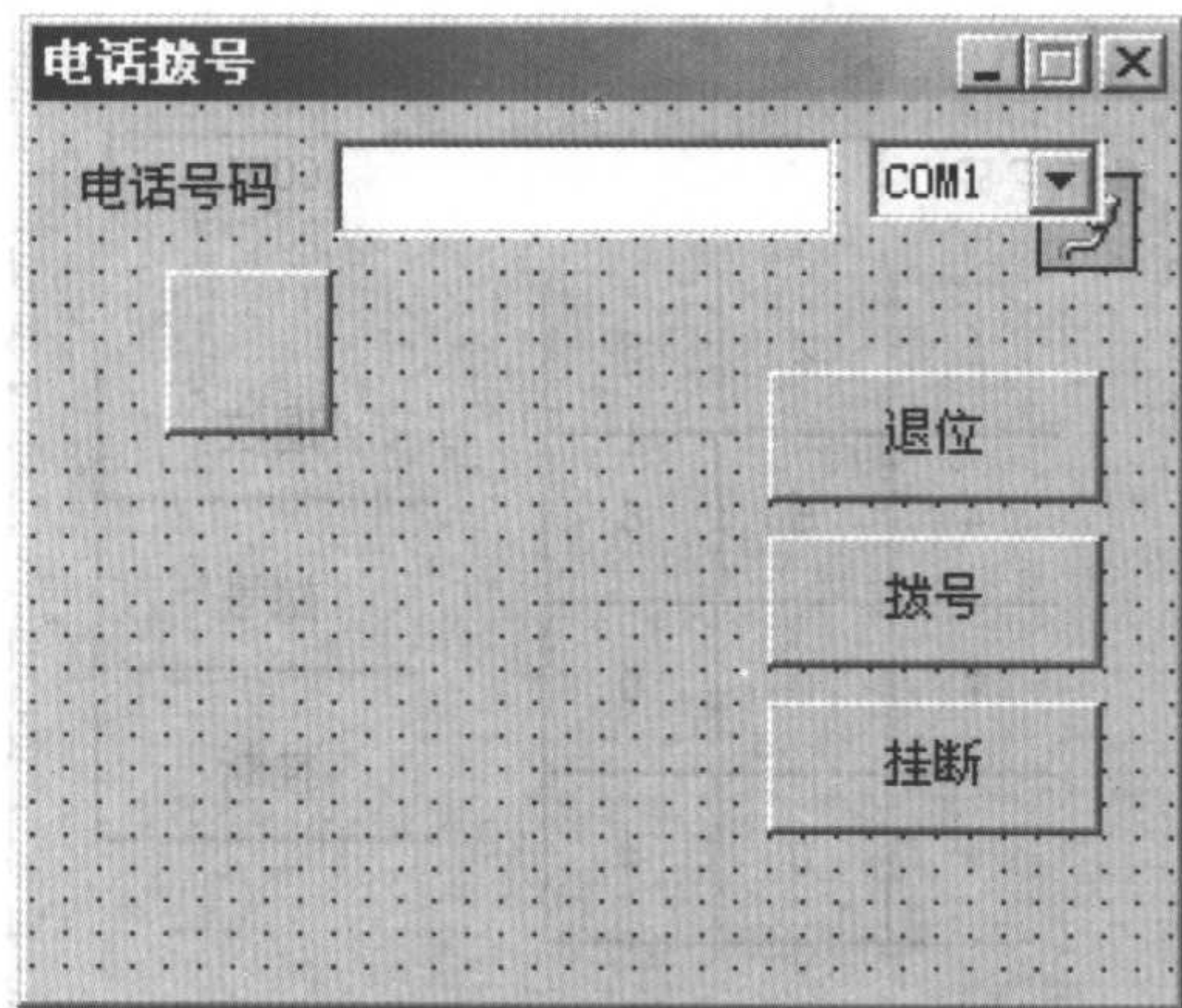


图 9-24 “电话拨号.e”设计界面

设置组合框的“类型”属性为“不可编辑下拉方式”，“列表项目”属性中手工添加 COM1、COM2、COM3、COM4 四个项目；在项目数值中添加 1、2、3、4 四个数值。分别代表电脑的四个串口。将其“现行选中项”属性设为 0，默认取 COM1。分别设置标签与按钮的标题，其中一个按钮标题为空，是用来动态创建电话号码按键的。

需要定义一个程序集变量数组“键盘”用来保存动态创建的电话按键，数组类型为按钮，成员数为 12。

窗口程序集名	备注		
窗口程序集1			
变量名	类型	数组	备注
键盘	按钮	12	

然后动态建立数字按钮，其中键盘按钮是原空文本按钮。

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
容器	整型			

```

--▶ 计次循环首 (12, 容器)
    复制窗口组件 (键盘按钮, 键盘 [容器])
    键盘 [容器].左边 = 键盘按钮.左边 + ((容器 - 1) % 3) × 键盘按钮.宽度
    键盘 [容器].顶边 = 键盘按钮.顶边 + ((容器 - 1) \ 3) × 键盘按钮.高度
    键盘 [容器].标题 = 多项选择 (容器, "1", "2", "3", "4", "5", "6",
        "7", "8", "9", "*", "0", "#")
    键盘 [容器].可视 = 真
-- 计次循环尾 ()
    号码框.获取焦点 ()
    
```

运行效果如图 9-25 所示。



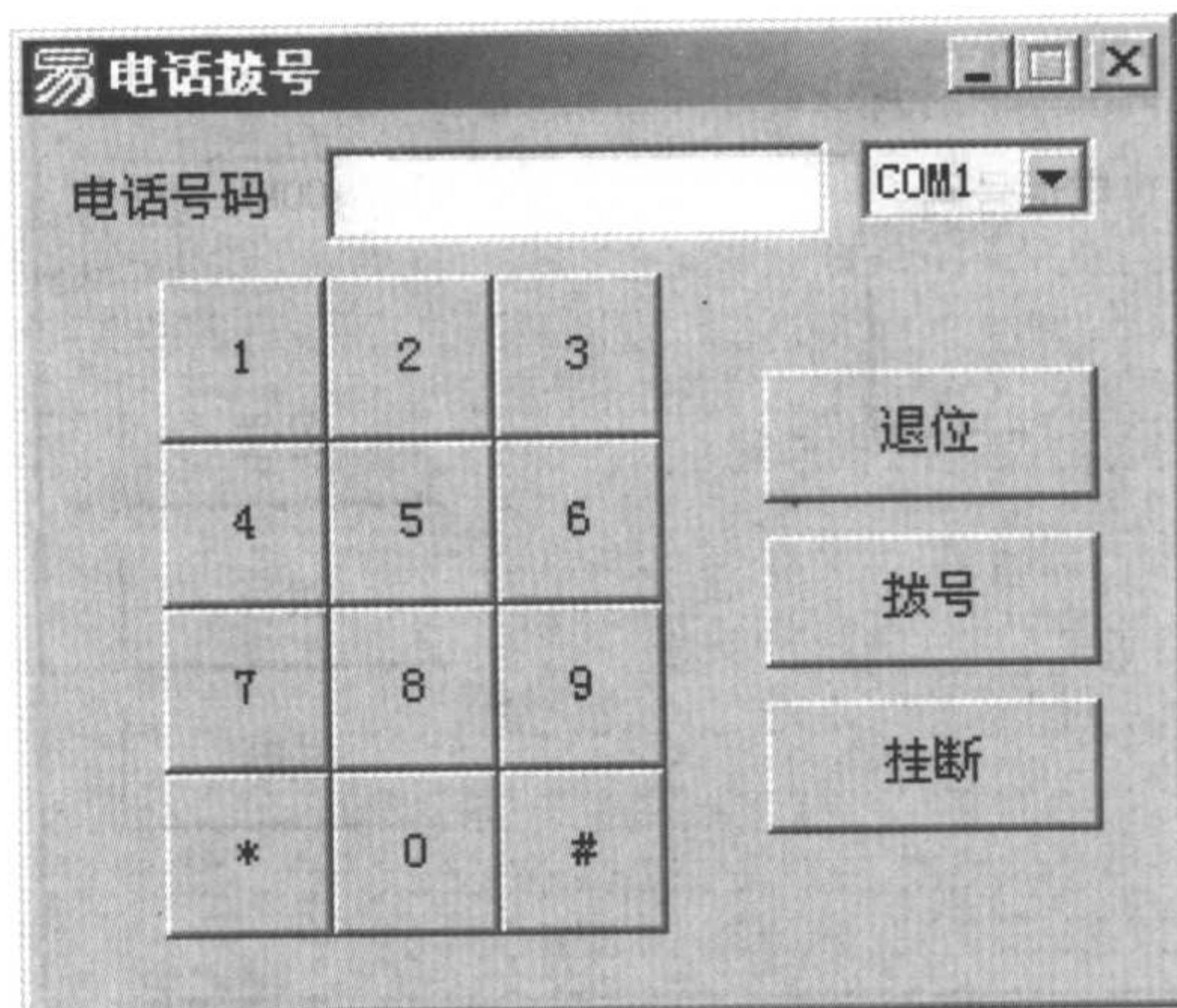


图 9-25 “电话拨号.e”运行界面

现在，对动态按钮的被单击事件做处理，该事件会自动被键盘按钮所捕获。

子程序名	返回值类型	公开	备注
键盘按钮_被单击			

变量名	类型	静态	数组	备注
临时按钮	按钮			
整数容器	整数型			

临时按钮 = 取事件组件 ()

--- 计次循环首 (12, 整数容器)

--- 如果真 (临时按钮 = 键盘 [整数容器])  
    号码框.加入文本 (临时按钮.标题)

--- 计次循环尾 ()

退位按钮被单击事件处理。

子程序名	返回值类型	公开	备注
_退位_被单击			

号码框.内容 = 取文本左边 (号码框.内容, 取文本长度 (号码框.内容) - 1)

拨号按钮被单击事件处理。

子程序名	返回值类型	公开	备注
_拨号_被单击			

端口1.端口号 = 组合框1.取项目数值 (组合框1.现行选中项)

--- 如果真 (端口1.启动 () = 假)

    信息框 (“初始化端口失败”, 0, )

    返回 ()

--- 如果真 (端口1.发送数据 (“ATDT” + 号码框.内容 + #换行符) = 假)

    信息框 (“拨号失败”, 0, )



挂断按钮被单击事件处理。

子程序名	返回值类型	公开	备注
_挂断_被单击			

端口1. 发送数据 (“ATH” + #换行符)  
 端口1. 停止 ()

在拨号按钮与挂断按钮的事件处理中，会看到在发送的数据前端添加了“AT 指令”。“AT 指令”是调制解调器专用协议，每条指令均以 AT 开头，D 代表拨号、T 代表用音频方式，H 代表挂断。ATDT 电话号码代表以音频方式拨打此电话号码，ATH 代表挂断电话。“AT 指令”很丰富，具体内容请查阅相关资料。

只要计算机与各类单片机设备进行硬件数据通信，就必须了解其相应的协议。各类专用机都有自己专有的协议规范。比如用于远程视频监视专用设备的协议指令（云台控制），最基础的指令仅启动、关闭、拉近、放远、左转、右转、上抬、下放八条。

### 9.4.2 端口访问支持库

端口组件局限于串口方式，对于并口通讯则无能为力。端口访问支持库则实现了对并口通讯的处理，不过它是直接读写并口的端口地址来发送或接收数据，与端口组件访问方式不同。

在使用此支持库之前，必须在当前计算机上安装端口访问驱动程序 port95nt.exe，包括程序分发后的终端用户电脑也必须安装此驱动，否则执行命令会失败。随书光盘中已附带安装文件，也可到易语言官方网站下载（<http://www.dywt.com.cn/port95nt.exe>）。

对通信端口做读写操作需要了解通信方协议，如打印机指令协议、扫描仪指令协议等，具体硬件设备的通讯协议请与设备厂商联系索取。

下面看一个简单例程，请确认您已安装端口访问驱动程序 port95nt.exe。

打开随书光盘本章目录中的“并口调试.e”，按 F5 运行，运行效果如图 9-26 所示。

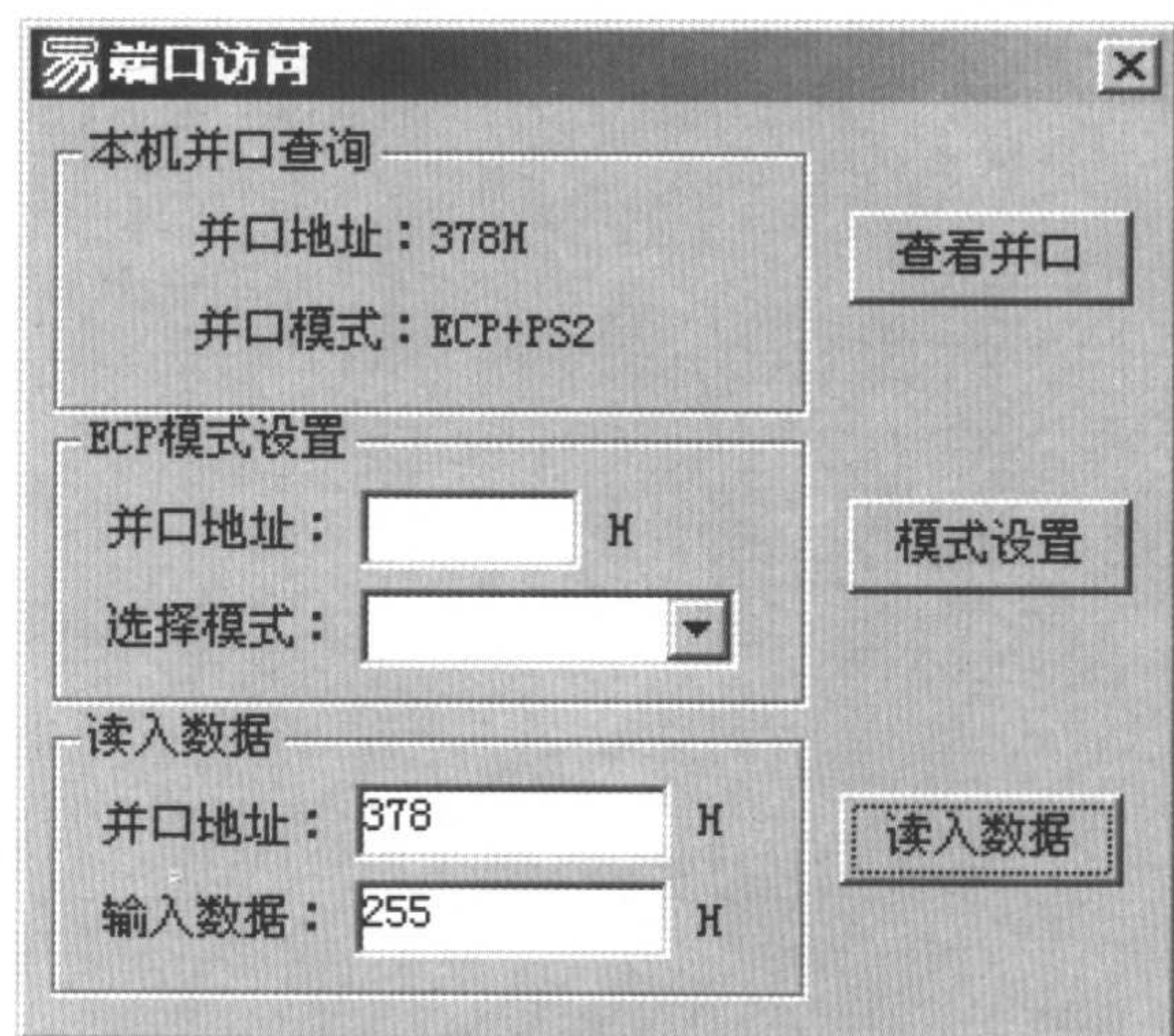


图 9-26 “并口调试.e”运行界面





首先点击“查看并口”按钮，得出并口地址，然后将取得的并口地址添入下面“并口地址”后的编辑框中，再点击“读入数据”按钮就可以读入数据。

点击不同的按钮可以实现不同的功能，具体代码请自行参考例程。在使用“模式设置”功能的时候一定要慎重，如果对硬件知识了解不够，有可能因地址冲突造成硬件损坏。建议在 CMOS 中手工设置，用程序读取测试为宜。

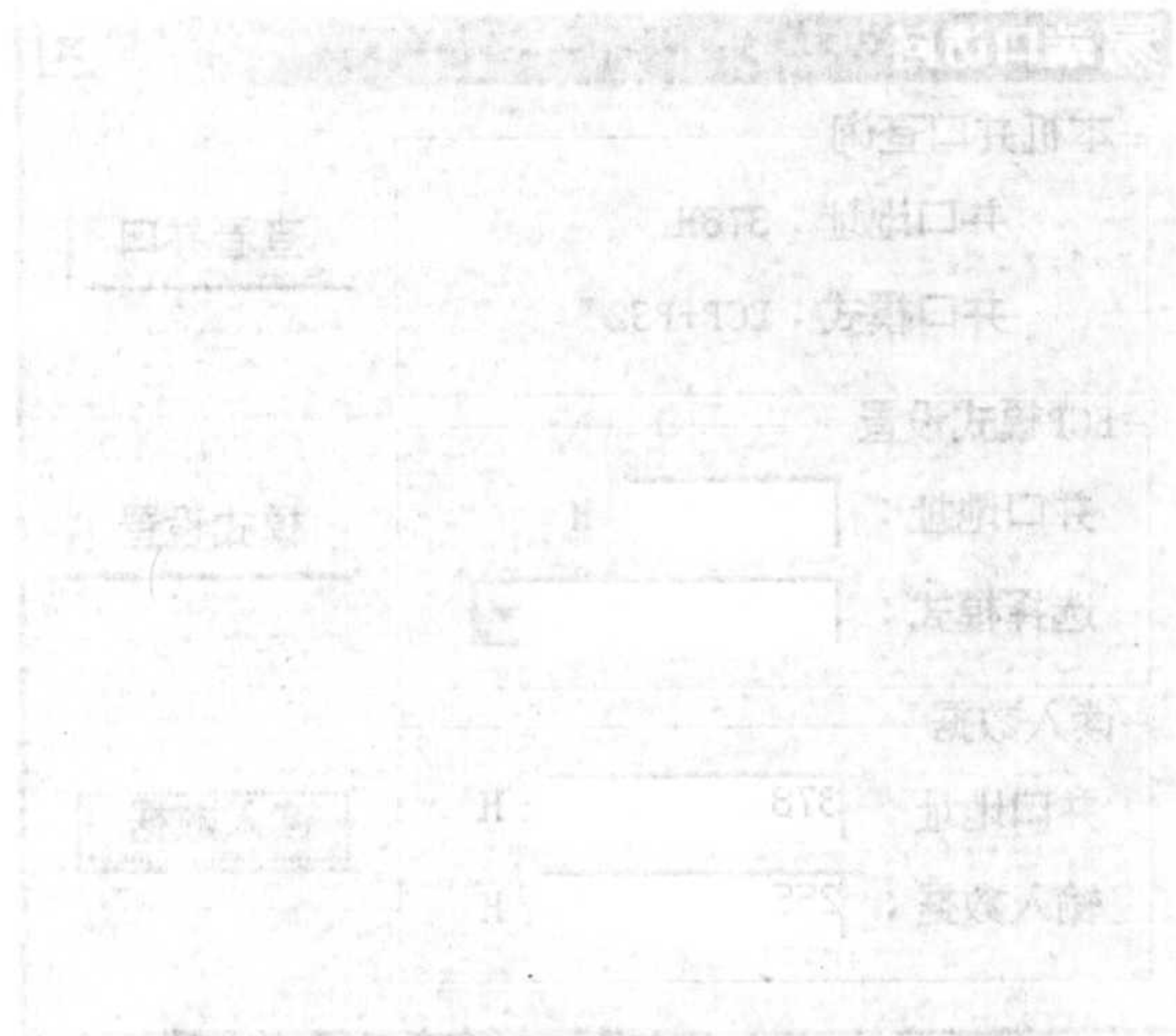
## 9.5 本章小结

硬件通信是数据通讯的基础，数据通讯则是网络应用的基础。作为网络型程序来说，我们不需要过多了解硬件物理层的操作原理，只需要基于现有各种协议或指令进行应用开发即可。

在实际开发工作中，有时候需要使用数据通讯组件实现标准协议功能，比如使用客户组件模拟 HTTP/FTP/SMTP/POP3 等协议，直接与服务器通讯，其指令全部以 ASCII 码传递，具体协议指令格式请查阅相关资料。

大家还可以进行以下练习：

1. 用数据报写一个双人聊天的程序。
2. 用服务器与客户组件编写一个多人网络聊天的程序。





## 第十章 系统控制

本章中首先介绍了一些易语言的“系统处理”类命令，后面还介绍了配置文件操作与注册表操作的方法。

易语言用“运行（）”命令调用外部程序运行。并且可以取得系统的一些设置信息，配置文件可方便在编程中为自己的程序保留信息设置等。易语言也可以方便地操作注册表，注册表的功能很多，不仅可以设置电脑软硬件，也可以用于保存注册信息等。

### 10.1 运行命令

“运行（）”命令属于易语言核心支持库的“系统处理”类命令。

“运行（）”命令运行指定的可执行文件或者外部命令，成功返回真，失败返回假。

实例代码如下：

```
运行（“D:\WINDOWS\System32\calc.exe”，假，）
```

上述代码可以实现运行系统中的计算器程序。

第二个参数设置为假，表示不等待外部程序运行完毕就直接返回，继续运行后续程序代码。第三个参数设置外部程序运行窗口的显示方式，参数值可以是以下常量之一：1、#隐藏窗口； 2、#普通激活； 3、#最小化激活； 4、#最大化激活； 5、#普通不激活； 6、#最小化不激活。如果省略本参数，默认为“普通激活”方式。

带参数调用外部程序方法如下：

```
运行（“NOTEPAD.EXE D:\WINDOWS\ntdtcsetup.log”，假，）
```

用“NOTEPAD.EXE”（记事本）程序打开“D:\WINDOWS\ntdtcsetup.log”文件。

```
运行（“explorer http://www.dywt.com.cn”，假，）
```

用“explorer”（资源管理器）程序打开“http://www.dywt.com.cn”网站，也可以打开本地磁盘的目录（文件夹）。代码如下：

```
运行（“explorer C:\”，假，）
```

运行DOS命令并立即返回。程序代码如下：

```
运行（“command.com /c copy”）
```

注意：外部程序与参数之间必须用空格分隔。如“command.com”和“/c copy”。





## 10.2 系统信息类命令

## 1. “取屏幕宽度 ()”

返回屏幕当前显示区域的宽度，单位为像素点。实例代码如下：

```
整数变量 = 取屏幕宽度 ()
```

## 2. “取屏幕高度 ()”

返回屏幕当前显示区域的高度，单位为像素点。实例代码如下：

```
整数变量 = 取屏幕高度 ()
```

打开随书光盘本章目录中例程“满天星.e”。

在“\_启动窗口\_创建完毕”事件子程序的程序代码如下：

```
_启动窗口.移动 (0, 0, 取屏幕宽度 (), 取屏幕高度 ())
```

```
画板1.移动 (0, 0, 取屏幕宽度 (), 取屏幕高度 ())
```

```
颜色数组 = { #白色, #蓝色, #绿色, #艳青, #红色, #黄色, #桃红, #嫩绿, #嫩黄,  
#天蓝, #紫色 }
```

这里用到的“移动 ()”命令应用十分广泛，可以方便地将组件移动和放大、缩小。当程序运行，“\_启动窗口\_创建完毕”事件子程序中的代码被执行，启动窗口被放大至屏幕大小，同时画板组件也放大至屏幕大小，将屏幕覆盖。

“\_时钟 1\_周期事件”事件子程序代码如下：

```
置随机数种子 0
```

```
画板1.画点 (取随机数 (0, 取屏幕宽度 ()), 取随机数 (0, 取屏幕高度 ()), 颜色数  
组 [取随机数 (1, 11)])
```

程序运行后，时钟组件的时钟周期事件（“时钟周期”属性值必须大于 0）被触发。在画板随机位置画彩色点。

## 3. “取鼠标水平位置 ()”

返回当前鼠标指针相对于屏幕左边的水平位置，单位为像素点。实例代码如下：

```
整数变量 = 取鼠标水平位置 ()
```

## 4. “取鼠标垂直位置 ()”

返回当前鼠标指针相对于屏幕顶边的垂直位置，单位为像素点。实例代码如下：

```
整数变量 = 取鼠标垂直位置 ()
```

例程“设置鼠标 1.e”，“\_时钟 1\_周期事件”事件子程序代码如下：

```
--- 如果真 (水平位置 ≠ 取鼠标水平位置 () 或 垂直位置 ≠ 取鼠标垂直位置 ())
```

```
水平位置 = 取鼠标水平位置 ()
```

```
垂直位置 = 取鼠标垂直位置 ()
```

```
编辑框1.内容 = 到文本 (水平位置)
```

```
编辑框2.内容 = 到文本 (垂直位置)
```



“水平位置”和“垂直位置”为静态变量，保存上一次鼠标停留的坐标位置。通过时钟组件定时对鼠标当前位置进行检查，如果鼠标位置发生了变化（即鼠标被移动），就保存最新的鼠标位置，并用编辑框将此位置坐标显示出来。

## 10.3 配置文件和注册表

### 10.3.1 配置文件

配置文件是操作系统或应用程序用来保存对软件或硬件各类设置信息的文件。系统配置文件主要包括 System.ini 和 Win.ini，其中保存了系统登录时硬件和软件的各种初始化信息，以便系统建立符合要求的工作环境。一般配置文件的后缀名为“.INI”，以文本方式保存。结构如图 10-1 所示。

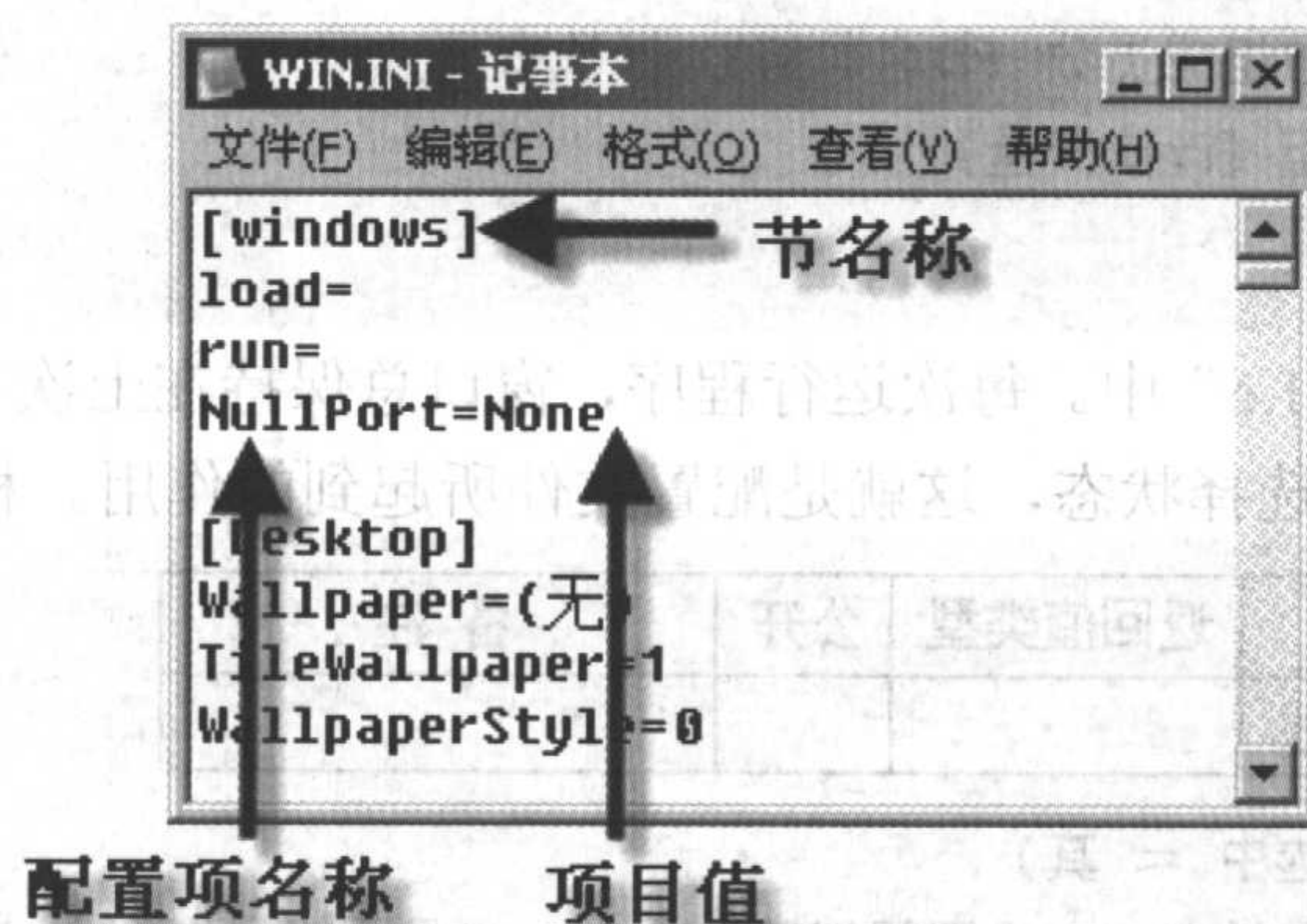


图 10-1 配置文件结构

一个配置文件可以有多个节，节与节之间由空行分隔。一个节可以有多个配置项，每个配置项可能有项目值也可能为空。

在易语言中提供了三个与读写配置文件相关的命令。

1. “写配置项 ( )”将指定文本内容写入指定配置项中，或者删除指定的配置项或节。如果指定配置文件不存在，将会自动创建。成功返回“真”，失败返回“假”。

实例代码如下：

逻辑变量=写配置项 (“C:\配置文件.ini”，“节名称”，“配置项名称”，“项目值”) 运行后所生成的新配置文件的内容如图 10-2 所示。

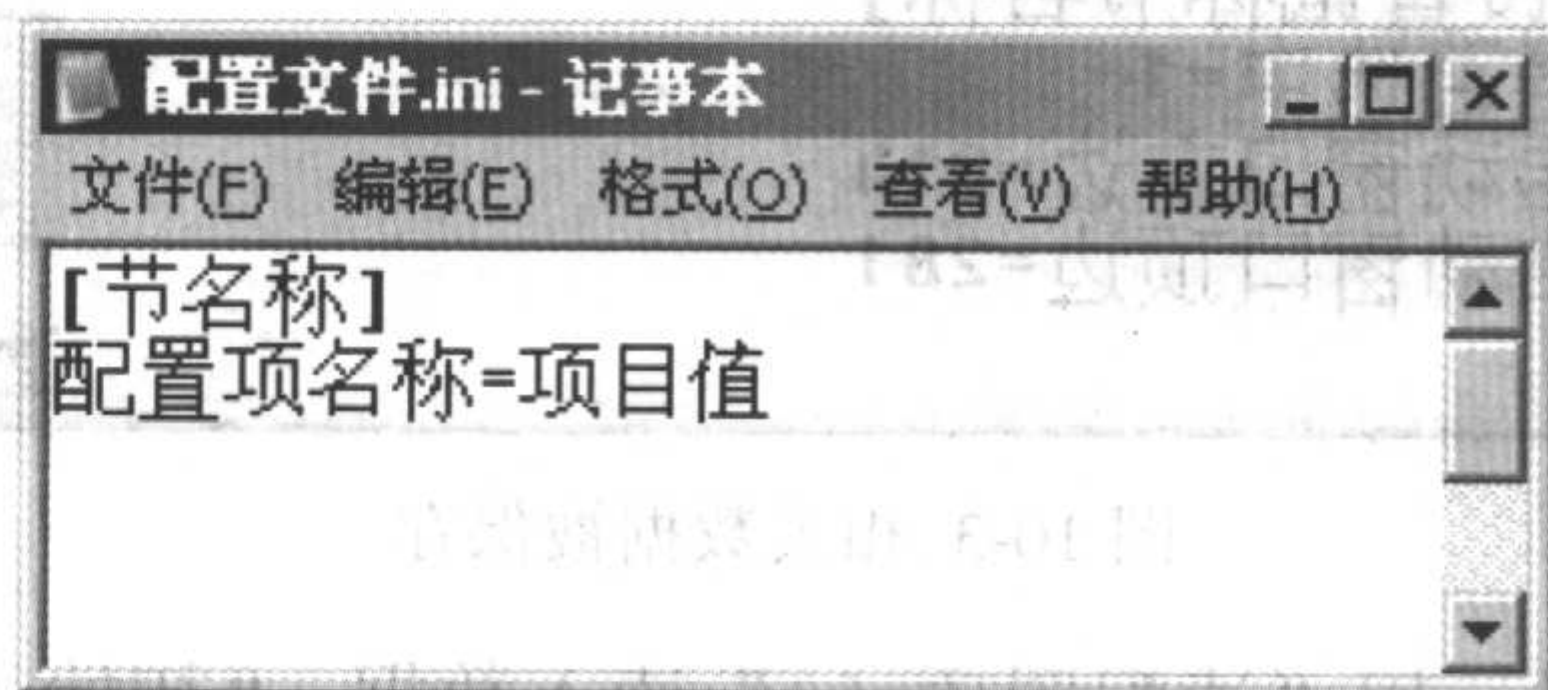


图 10-2 生成文件





2. “读配置项 ( )” 读取指定配置文件中指定配置项的内容。

文本变量=读配置项 (“c:\配置文件.ini”, “节名称”, “配置项名称”, “默认项目值”)

读出的“项目值”将保存在文本变量中。如果指定配置项不存在, 将返回“默认项目值”。如果指定配置项不存在且“默认项目值”参数被省略, 将返回空文本。

3. “取配置节名 ( )” 返回指定配置文件中所有已存在节名的文本数组。

实例代码如下:

变量名	类型	静态	数组	备注
节名数组	文本型		0	
计次变量	整数型			
节名	文本型			

节名数组 = 取配置节名 (“c:\配置文件.ini”)

--- 计次循环首 (取数组成员数 (节名数组), 计次变量)

节名 = 节名数组 [计次变量]

--- 计次循环尾 0

在例程“设置鼠标 1.e”中。每次运行程序, 窗口总保持在上次关闭时的位置, 选择框也保持在上次关闭时的选择状态, 这就是配置文件所起到的作用。相关程序代码如下:

子程序名	返回值类型	公开	备注
__启动窗口_将被销毁			

--- 如果 (选择框1.选中 = 真)

--- 写配置项 (配置文件名, “设置鼠标节名称”, “是否选中”, “1”)

--- 写配置项 (配置文件名, “设置鼠标节名称”, “是否选中”, “0”)

写配置项 (配置文件名, “设置鼠标节名称”, “启动窗口左边”, 到文本  
(\_\_启动窗口.左边))

写配置项 (配置文件名, “设置鼠标节名称”, “启动窗口顶边”, 到文本  
(\_\_启动窗口.顶边))

程序结束之前, 用“写配置项 ( )” 命令将相关数据保存到配置文件中。文件如图 10-3 所示。

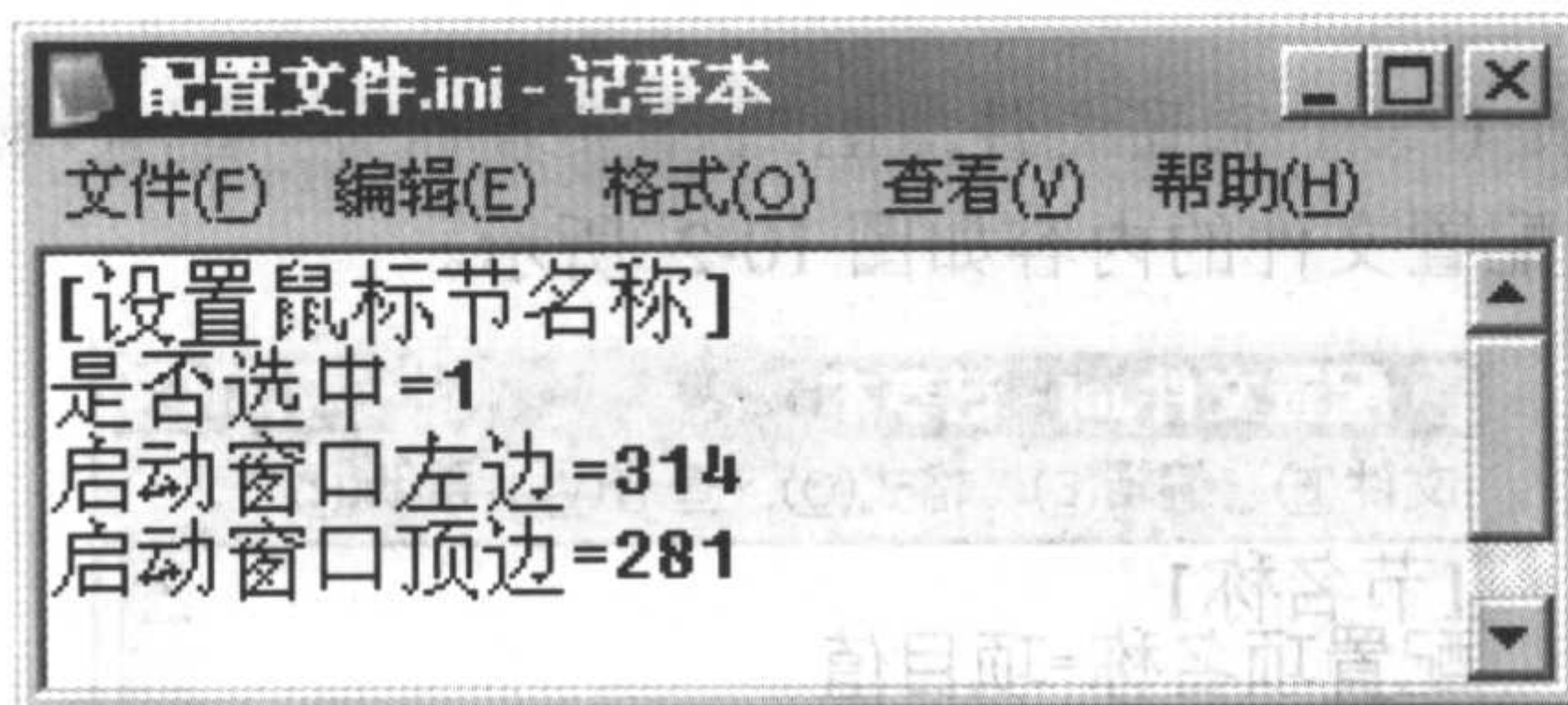


图 10-3 相关数据被保存

当程序重新被运行时, 用“读配置项 ( )” 读入数据。“初始子程序”中相关程序代码如下:



```

如果 (读配置项 (配置文件名, “设置鼠标节名称”, “是否选中”
, “1”) = “1”)
选择框1.选中 = 真
选择框1.选中 = 假
窗口左边 = 到数值 (读配置项 (配置文件名, “设置鼠标节名称”, “启动
窗口左边”, ))
窗口顶边 = 到数值 (读配置项 (配置文件名, “设置鼠标节名称”, “启动
窗口顶边”, ))
启动窗口.移动 (窗口左边, 窗口顶边, , )

```

当然,有些应用程序基于简便、安全或空间占用等方面的原因,不使用读写配置命令读写标准配置文件,而是采取通过文件读写命令读写自定义格式的配置文件,同样也可以达到配置程序的目的。

### 10.3.2 调用系统配置工具

“运行( )”命令不但可以调用系统中可执行程序 and 文件(10.1 中介绍过多种使用方法),而且还可以调用控制面板及其所包含的系统配置工具。例程“打开控制面板.e”,程序界面如图 10-4 所示。

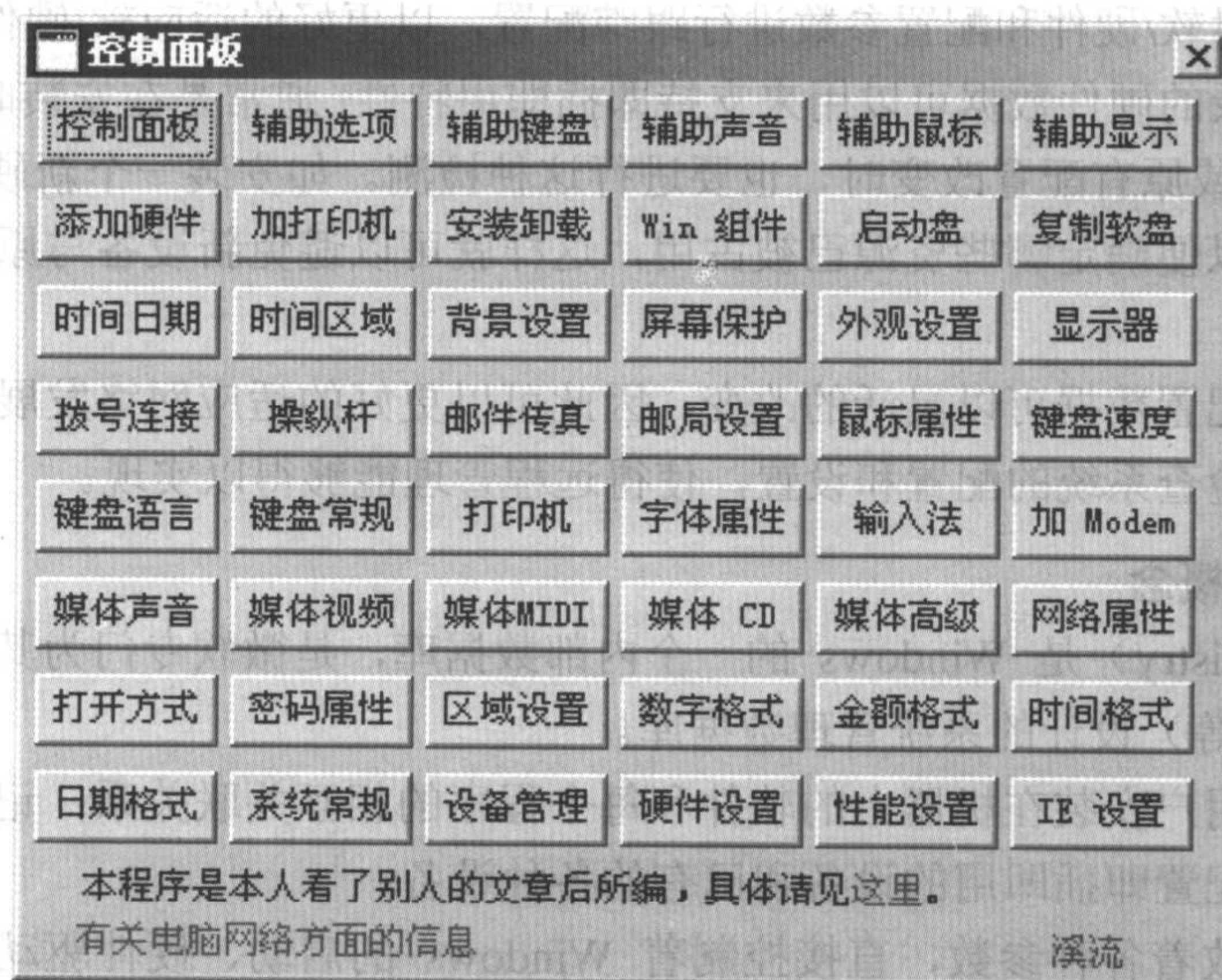


图 10-4 程序界面

通过运行带有不同参数的命令,可以调用相应的系统内部配置工具。代码格式如下:

```
运行 (“rundll32.exe shell32.dll,Control_RunDLL”, 假, )
```

上行代码可调用控制面板,如图 10-5 所示。





图 10-5 系统控制面板

### 10.3.3 注册表

从功能上来说，注册表相对于 INI 文件具有以下优点：

注册表允许对软/硬件和配置参数进行跟踪配置，以更好的适应软/硬件；

注册表中登录的硬件数据可以用来支持即插即用特性；通常是在安装时进行这种检测，但 Windows 启动或原有配置改变时，也要进行这种检测。如安装一个新硬件时，Windows 将检查注册表，以便确定哪些资源已被占用，这样就可以避免新设备与原有设备之间的资源冲突。

注册表中的配置数据可以灵活的改变，因此可以更好的适应网络发展。管理人员和用户可以在网络上检查系统的配置和设置，使得远程管理能够得以实现。

#### 1. 注册表的概念

注册表 (Registry) 是 Windows 的一个内部数据库，是微软专门为其 32 位操作系统 (Windows 系列等) 设计的系统管理数据库。

注册表记录用户安装在机器上的软件和每个程序的相互关联关系；记录计算机的硬件配置，包括自动配置即插即用的设备和已有的各种设备。

注册表中存放着各种参数，直接控制着 Windows 的启动、硬件驱动程序的装载以及一些 Windows 应用程序的运行，在整个系统中起着核心作用。它保存的信息包括以下几个方面内容：

(1) 软、硬件的有关配置和状态信息。注册表中保存有应用程序和资源管理器外壳的初始条件、首选项和卸载数据。

(2) 联网计算机整个系统的设置和各种许可，文件扩展名与应用程序的关联，硬件部件的描述、状态和属性。

(3) 电脑性能记录和其他底层的系统状态信息，以及其他数据。

如果注册表受到了破坏，轻则使 Windows 的启动过程出现异常，重则可能会导致整



个系统的完全瘫痪，因此要正确地认识和使用注册表。特别是要经常备份，以备系统出现问题能及时恢复正确的注册表配置。

总的来说，注册表实际上就是一个以层次结构保存和检索的复杂数据库，包含了应用程序和系统软硬件的全部配置、初始化信息以及其他重要数据。

## 2. 注册表的结构

在注册表中，所有的数据都是通过一种树状结构以键和子键的方式组织起来的，就像磁盘文件系统的目录结构一样。每个键都包含了一组特定的信息，每个键的键名都是和它所包含的信息相关联的。下面列出了注册表树最顶层的六个主键（也称根键）所代表的含义，系统规定这六个主键是不允许修改名称和删除的。

### (1) HKEY\_CLASSES\_ROOT

管理文件系统。在这个根键中保存着操作系统中所有已登记文件类型的扩展名，及打开该文件所要调用的程序等信息。

### (2) HKEY\_CURRENT\_USER

管理系统当前的用户信息。在这个根键中保存了本地计算机中存放的当前登录的用户信息，包括用户登录用户名和暂存的密码。在用户登录 Windows 时，其信息从 HKEY\_USERS 中相应的项拷贝到 HKEY\_CURRENT\_USER 中。

### (3) HKEY\_LOCAL\_MACHINE

管理当前系统硬件配置。在这个根键中保存了本地计算机硬件配置数据，此根键下的子关键字数据内容保存在 SYSTEM.DAT 中，或者远程计算机可访问的一组键中。

### (4) HKEY\_USERS

管理系统的用户信息。在这个根键中保存了存放在本地计算机口令列表中的用户标识和密码列表。同时每个用户的预配置信息都存储在 HKEY\_USERS 根键中。HKEY\_USERS 是远程计算机中访问的根键之一。

### (5) HKEY\_CURRENT\_CONFIG

管理当前用户的系统配置。在这个根键中保存着定义当前用户桌面配置(如显示器等等)的数据，该用户使用过的文档列表 (MRU)，应用程序配置和其他有关当前用户的 Windows 中文版的安装的信息。

### (6) HKEY\_DYN\_DATA

管理系统运行数据。该根键只在 Windows98 下存在，其中保存了系统在运行时的动态数据，此数据在每次显示时都是变化的，因此，此根键下的信息没有放在注册表中。

## 3. 注册表项键值的数据类型

### (1) 字符串值 (REG-SZ)

在 Windows 的注册表中，表示文件的描述和硬件的标识等信息一般都用字符串值。字符串值由字母和数字组成，它的最大长度不能超过 255 个字符。通过键、键值就组成了一种键值项数据，这就相当于 Win.ini、System.ini 文件中每个小节下面的设置行一样的道理。如“Windows 标准”，是一串文字或词组。注册表总是在引号内显示字符串。





## (2) 二进制值 (REG-BINARY)

在 Windows 的注册表中，二进制值是没有长度限制的，可以是任意个字节长。在注册表编辑器中，二进制以十六进制的方式显示出来。如 F5FFFFFF00000000，是有限制的二进制数值，用 16 进制显示。

## (3) DWORD 值 (REG-DWORD)

在 Windows 的注册表中，DWORD 值是一个 32 位（双字节长）长度的数值。在注册表编辑器中，系统以十六进制的方式显示 DWORD 值。如 c0c0c0，以 8 位 16 进制数表示的数。

## 4. 注册表结构图解

注册表是一个树状分层的数据库。从物理上讲，它是 System.dat 和 User.dat 两个文件；从逻辑上讲，它是用户在注册表编辑器中看到的配置数据。结构如图 10-4 所示。

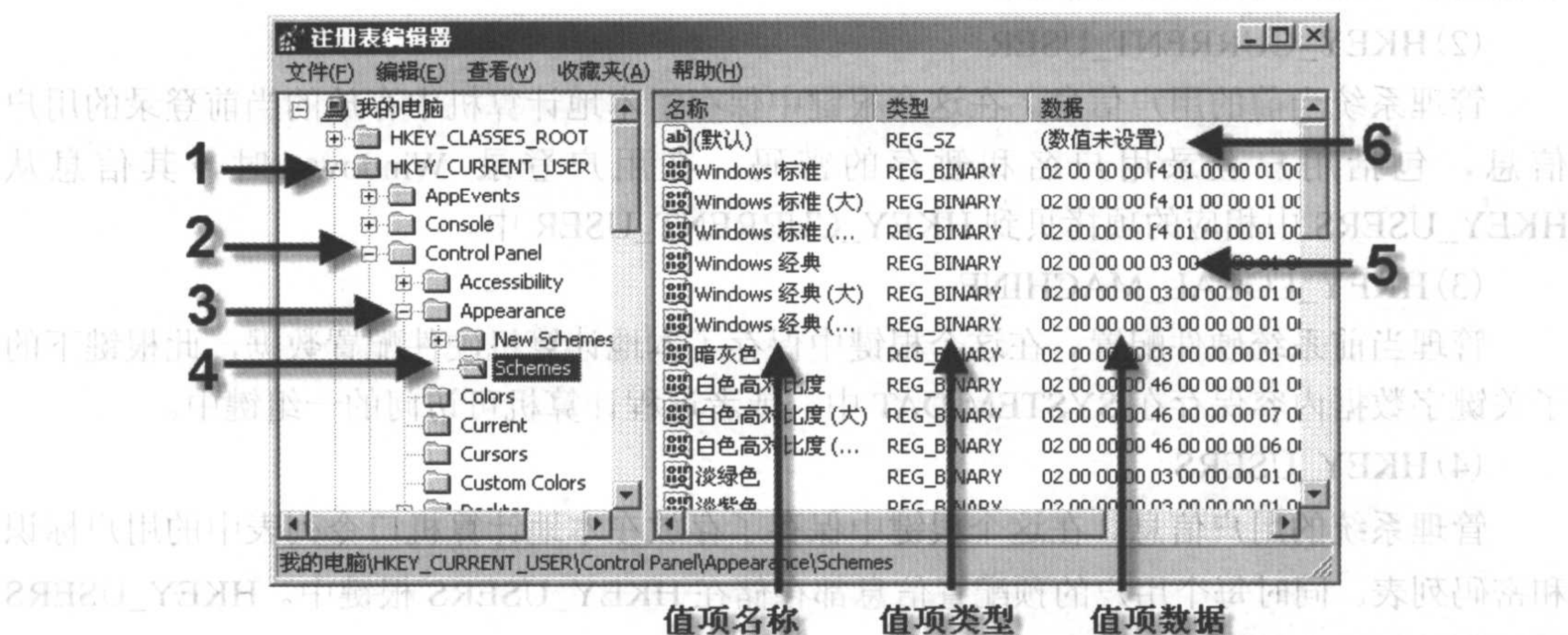


图 10-4 注册表编辑器

① HKEY：“根键”。Windows 将注册表根键以“HKEY\_name”的方式命名，它意味着某一键的句柄。

② key(键)：它包含了附加的文件夹和一个或多个值。

③ subkey(子键)：在某一个键(父键)下面出现的键(子键)。

④ branch(分支)：代表一个特定的子键及其所包含的一切。一个分支可以从每个注册表的顶端开始，但通常用以说明一个键和其所有内容。

⑤ value entry(值项)：带有一个名称和一个值的有序值。每个键都可包含任何数量的值项。每个值项均由三部分组成：名称，数据类型，数据。

名称：不包括反斜杠的字符、数字、代表符、空格的任意组合。同一键中不可有相同的名称。

数据类型：包括字符串、二进制、双字三种。

数据：值项的具体值，它可以占用到 64KB。

⑥ Default(缺省值)：每一个键至少包括一个值项，称为缺省值(Default)，它总是一个字符串。



### 5. HKEY\_LOCAL\_MACHINE 键的重要子键及其作用

①HKEY \_ LOCAL \_ MACHINE \ software \ microsoft \ windows \ currentVersion \ uninstall 保存 Windows 系统中已经安装了的 Windows 应用程序卸载信息。

②HKEY\_LOCAL\_MACHINE\system\currentControl-Set\control\keyboard Layouts 保存 Windows 中键盘使用的语言以及各种中文输入法的信息。

③ HKEY\_LOCAL\_MACHINE\software\microsoft\windows\currentVersion\explorer\usershell folders 保存计算机中个人文件夹、收藏夹的路径。

④HKEY\_LOCAL\_MACHINE\system\CurrentControl-Set\services\class 保存控制面板-增添硬件设备-设备类型目录, 全面管理你的硬件信息。

⑤HKEY\_LOCAL\_MACHINE\software\microsoft\win-dows\currentVersion\run 保存由控制面板设定的计算机启动时运行程序的名称, 其图标显示在任务条右边。这也是经常修改和用到的一个目录。

⑥HKEY\_LOCAL\_MACHINE\software\microsoft\windows\currentVersion\Policies\ Ratings 保存了 IE 的“安全”\“分级审查”中设置的口令(数据加密), 若遗忘了口令, 删除 Ratings 中的数据即可解决问题。

⑦HKEY\_LOCAL\_MACHINE\software\microsoft\windows\currentVersion\explorer\ desktop\ name Space 保存桌面中特殊的图标, 回收站、收件箱、网上邻居等等。

### 6. HKEY\_USERS 键的重要分支

①HKEY\_USERS\Default\so..\microsoft\windows\current-Version\explorer\RunMRU 保存“开始 \ 运行”中运行的程序列表信息。清除文档菜单时该分支将被清空。

②HKEY\_USERS\Default\software\microsoft\internet explorer\typeURLs 保存 IE4.0 浏览器地址栏中输入的 URL 地址列表信息。清除文档菜单时它将被清空。

③HKEY\_USERS\Default\Software\microsoft\windows\current-Version\explorer\ RecentDocs 保存最近使用的十五个(数目是可以修改的)文档的快捷方式, 清除文档菜单时将被清空。

④HKEY\_USERS\default\software\microsoft\windows\currentVersion\applets 保存 Windows 应用程序的记录数据信息。

上面介绍了 Windows 的注册表的结构和重要的信息, 这对于修改和操作注册表是非常有用的。

许多商品化的软件或专业化的软件在您的机器上首次安装的时候都会通过改写注册表来完成软件的正确安装运行, 梦想成为编程高手的您当然需要掌握读写注册表这一技术。利用好注册表会为您的应用程序增色不少。

在编程时一般只是用到 HKEY\_CURRENT\_USER 和 HKEY\_LOCAL\_MACHINE 这两个根键, 因为与应用程序相关的数据仅存在于这两个根键下。

### 7. 如何通过程序来操作注册表

在易语言中注册表的根键被设置为常量, 对照如下:

- (1)#根类.....HKEY\_CLASSES\_ROOT
- (2)#现行设置.....HKEY\_CURRENT\_CONFIG
- (3)#现行用户.....HKEY\_CURRENT\_USER





(4)#本地机器.....HKEY\_LOCAL\_MACHINE

(5)#所有用户.....HKEY\_USERS

在易语言中作为常量使用的实例代码如下:

文本变量 = 取文本注册项 (3, "Environment\include", "123")

第一个参数的3 代表的是常量名为“#现行用户”的根目录。

“取文本注册项()”命令在 Windows 注册表中返回指定的文本类型注册表项值。如欲读取注册项默认值,请在项目名后加“\”号,如“test\”。

实例代码如下:

文本变量 = 取文本注册项 (#现行用户, "Environment\include", "123")

第二个参数指全路径位置的项目值,必须是 REG-SZ 类型。REG-SZ 类型是文本型。如图 10-5 所示。

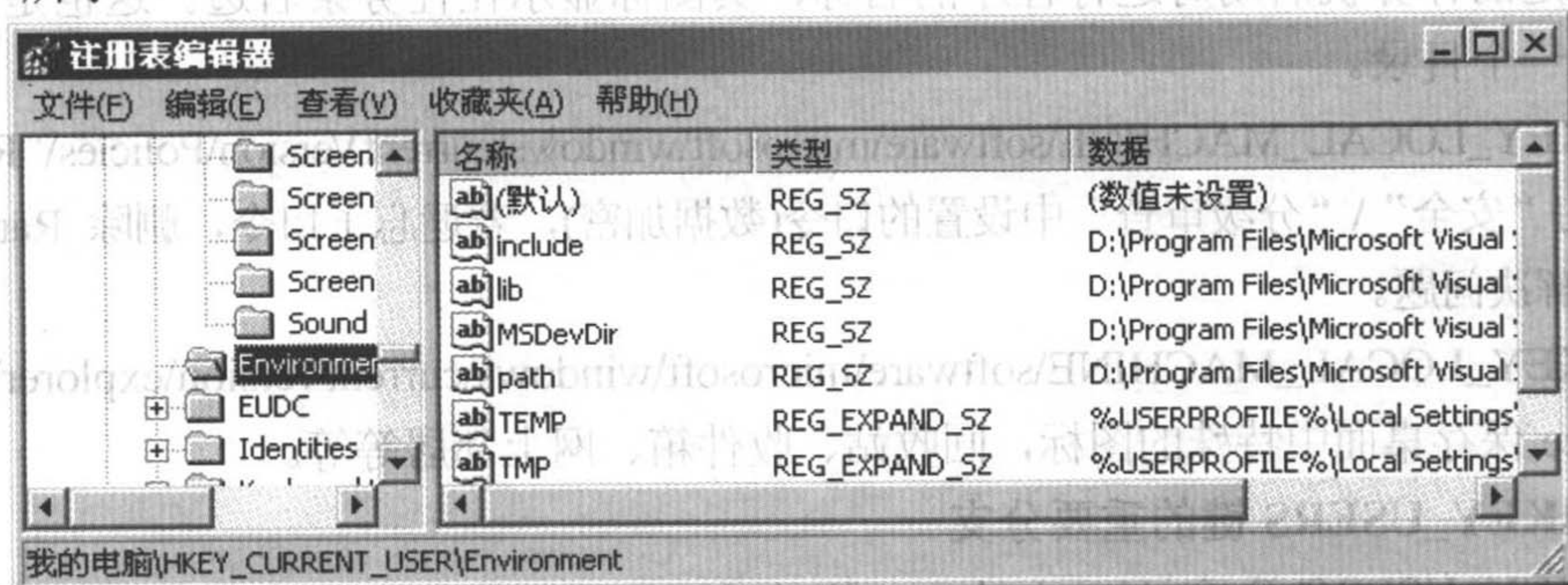


图 10-5 文本型注册表项值

## 8. 注册表命令

(1)“取数值注册项()”命令

该命令在 Windows 注册表中返回指定的数值类型注册表项值。如欲读取注册项默认值,请在项目名后加“\”号,如“test\”。

实例代码如下:

数值变量 = 取数值注册项 (3, "Software\Microsoft\Internet Explorer\Main\NoJITSetup", 123)

在注册表中, REG-DWORD 类型是数值型。如图 10-6 所示。



图 10-6 数值型注册表项



## (2) “取字节集注册项 ( )” 命令

该命令在 Windows 注册表中返回指定的字节集类型注册表项值。如欲读取注册项默认值，请在项目名后加 “\” 号，如 “test\”。

实例代码如下：

```
字节集变量 = 取字节集注册项 (# 现行用户, “Software\Microsoft\Internet Explorer\Main\Do404Search”, )
```

在注册表中，REG-BINARY 类型是字节集型。如图 10-7 所示。

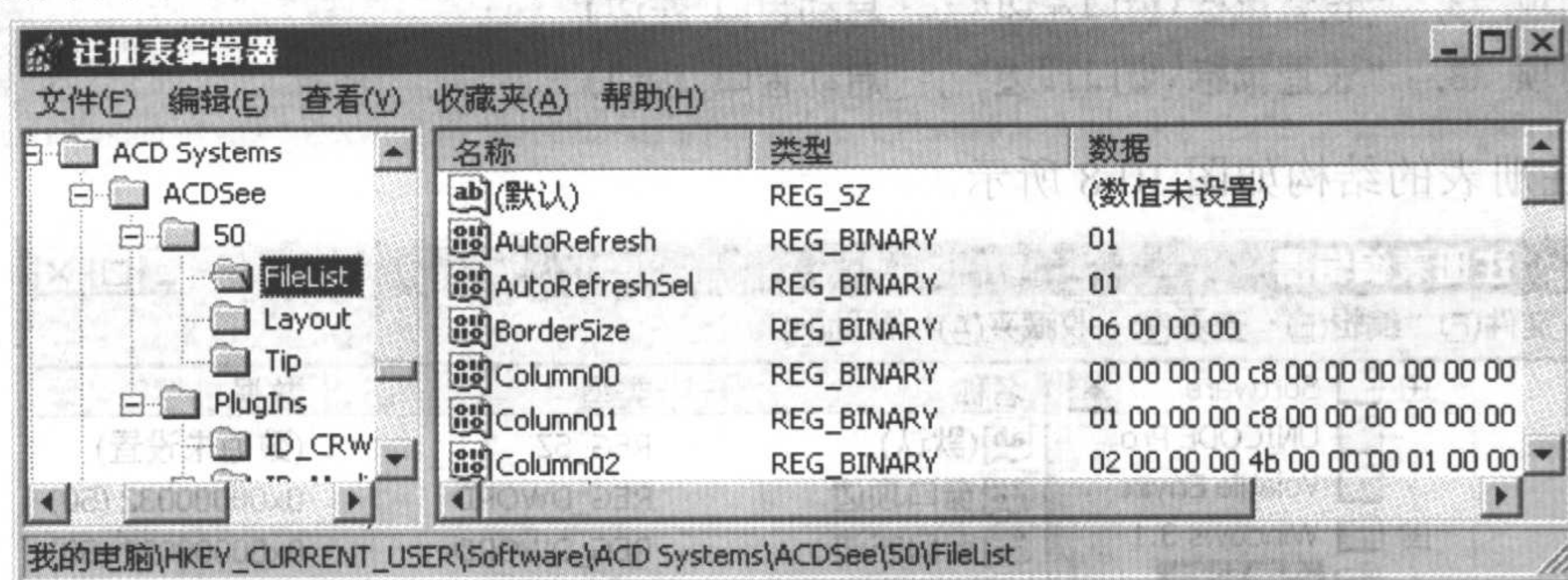


图 10-7 字节集型注册表项

## (3) “写注册项 ( )” 命令

该命令在 Windows 注册表中保存或建立指定的注册表项。如欲写入注册项默认值，请在项目名后加 “\” 号，如 “test\”。成功返回真，否则返回假。

实例代码如下：

```
逻辑变量 = 写注册项 (# 现行用户, “Software\Microsoft\Internet Explorer\Main\dj”, 555)
```

## (4) “删除注册项 ( )” 命令

该命令在 Windows 注册表中删除指定注册表项或注册表目录。如欲删除注册项默认值，请在项目名后加 “\” 号，如 “test\”。成功返回真，否则返回假。与 “删除注册表项” 命令不同的是本命令可以删除任意位置处的注册表项或目录。注意在删除目录之前必须先删除该目录下所有的项目。

实例代码如下：

```
逻辑变量 = 删除注册项 (# 现行用户, “Software\Microsoft\Internet Explorer\Main\dj” )
```

## (5) “注册项是否存在 ( )” 命令

如果指定注册表项存在，返回真，否则返回假。如欲检查注册项是否有默认值，请在项目名后加 “\” 号，如 “test\”。

实例代码如下：

```
逻辑变量 = 注册项是否存在 (# 现行用户, “Software\Microsoft\Internet Explorer\Main\Start Page” )
```

例程 “设置鼠标 2.e” 和前面的 “设置鼠标 1.e” 例程只是保存配置的方式不同。例程





“设置鼠标 2.e”将使用注册表保存程序的配置。其中写注册项()的相关程序代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_将被销毁			

```
--- 如果 (选择框1.选中 = 真)
--- 写注册项 (3, “设置鼠标\选择框选中”, “1”)
--- 写注册项 (3, “设置鼠标\选择框选中”, “0”)
写注册项 (3, “设置鼠标\窗口左边”, _启动窗口.左边)
写注册项 (3, “设置鼠标\窗口顶边”, _启动窗口.顶边)
```

写到注册表的结构如图 10-8 所示。

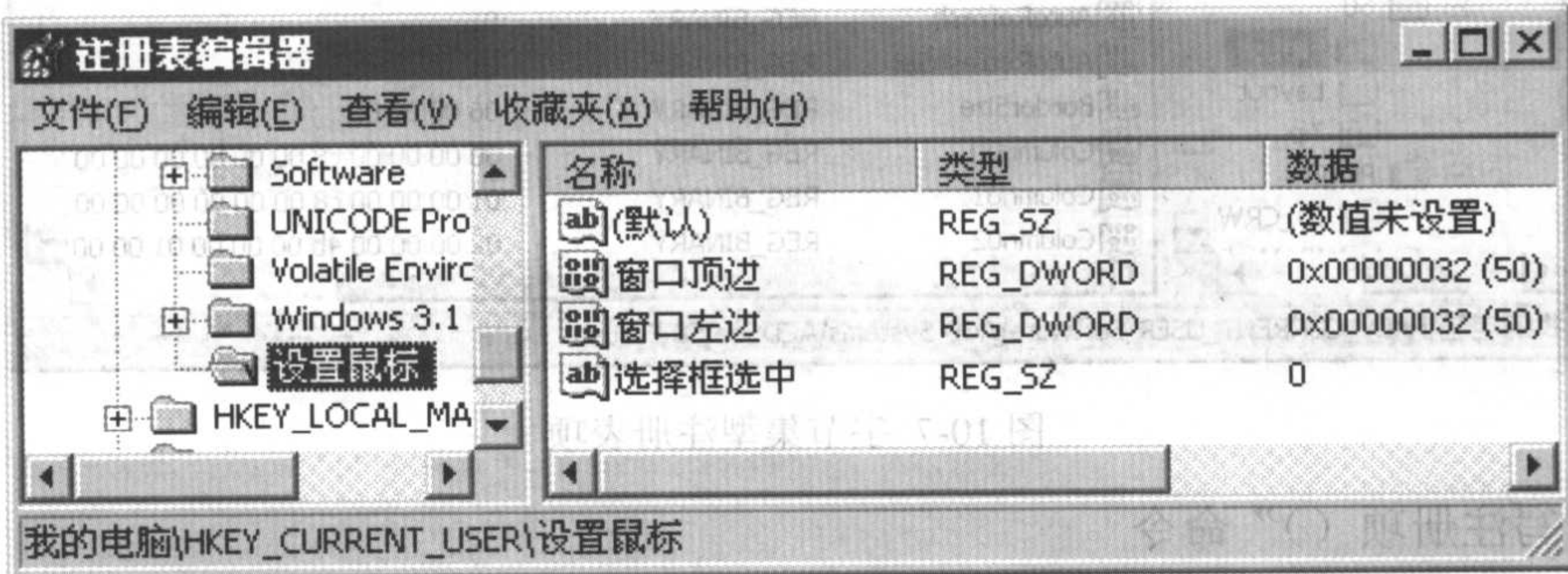


图 10-8 写出结果

程序运行时“初始子程序”代码如下：

```
--- 如果 (取文本注册项 (3, “设置鼠标\选择框选中”, ) = “1”)
--- 选择框1.选中 = 真
--- 选择框1.选中 = 假
_启动窗口.移动 (取数值注册项 (3, “设置鼠标\窗口左边”, ), 取数值注册项
(3, “设置鼠标\窗口顶边”, ), , )
```

由于注册表结构复杂，为了能对注册表的操作更加了解，下面介绍不同的写出方法。

写注册项 (3, “”, “aaaaa”)

写根键的默认项目值为“aaaaa”。如图 10-9 所示。

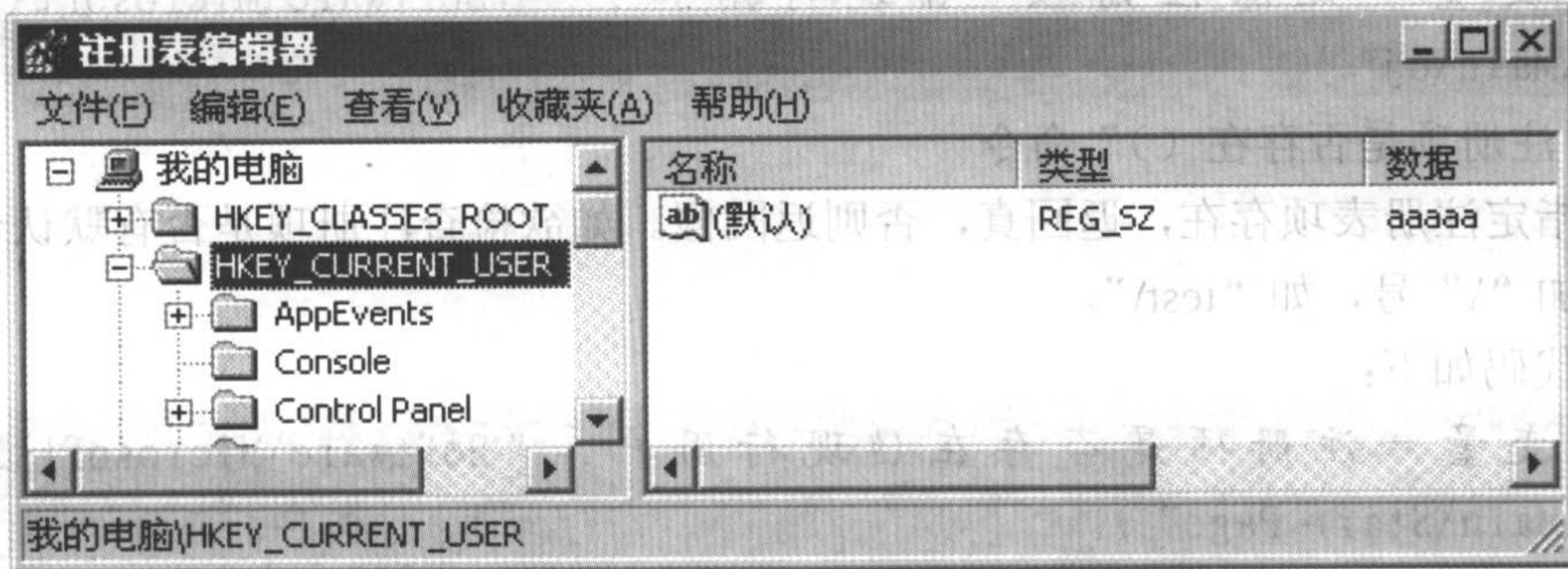


图 10-9 写根键默认项目值



写注册项 (3, “AAAAA”, “aaaaa”)

在根键下添加注册项“AAAAA”，其值为“aaaaa”。如图 10-10 所示。

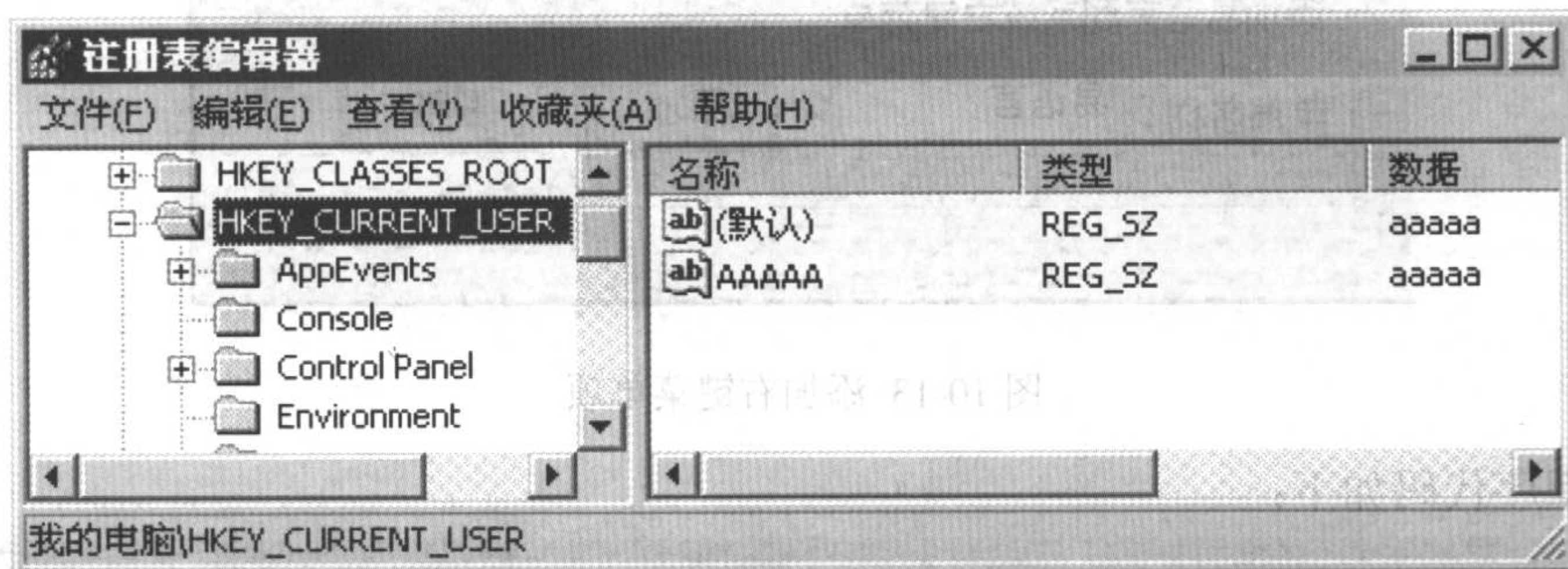


图 10-10 添加注册项

写注册项 (3, “AAAAA\”, “aaaaa”)

在根键下添加子键“AAAAA”，其默认项目值为“aaaaa”。如图 10-11 所示。

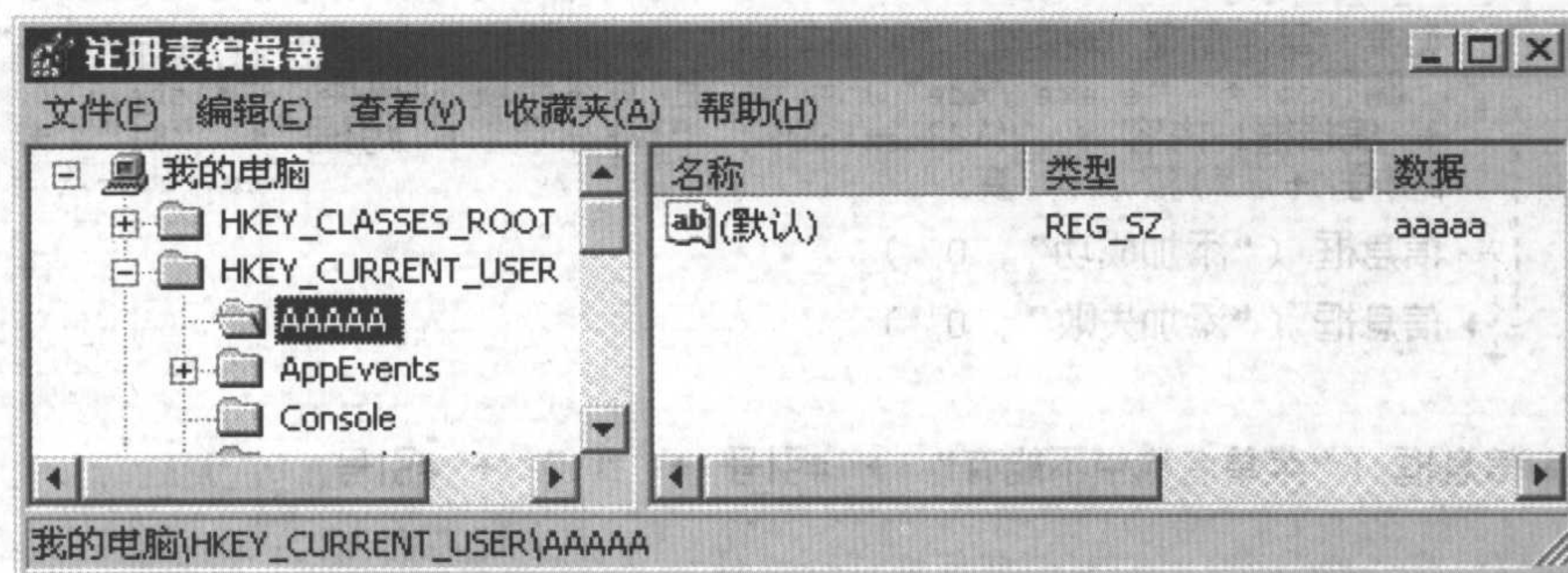


图 10-11 添加子键

写注册项 (3, “AAAAA \ BBBB”, “aaaaa”)

在子键“AAAAA”下添加注册项“BBBBB”，其值为“aaaaa”。如图 10-12 所示。

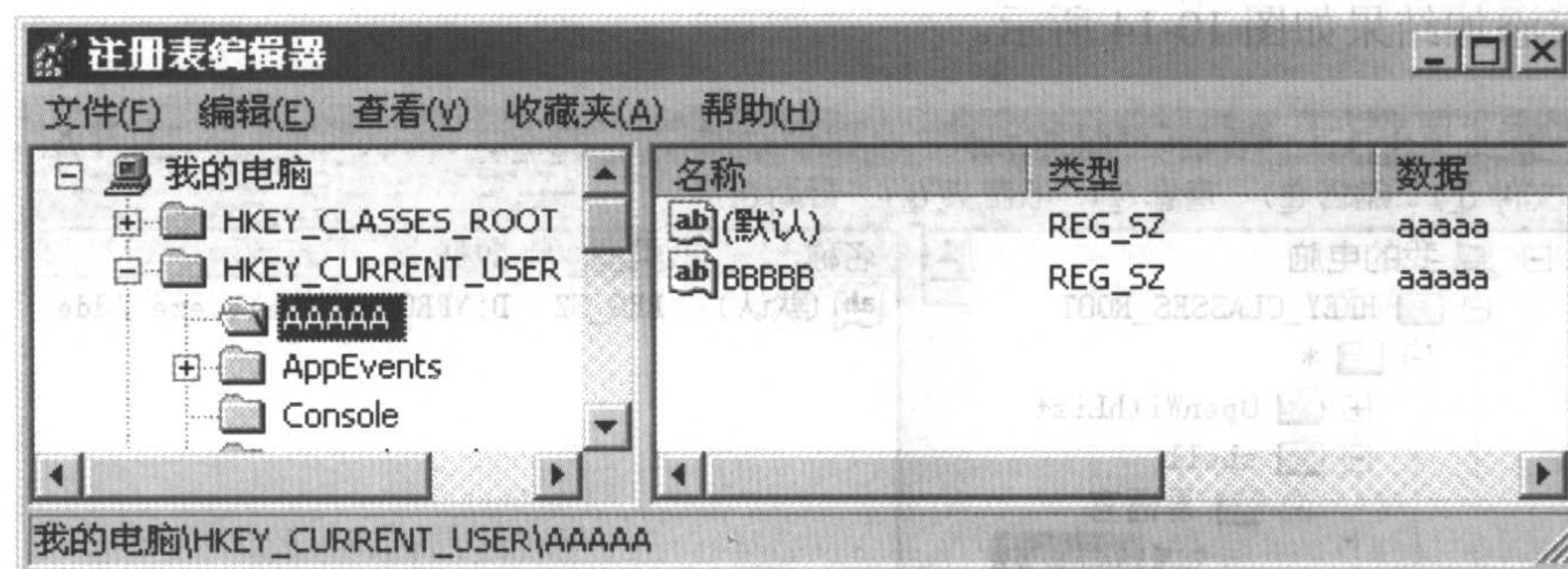


图 10-12 子键添加注册项

## 9. 添加系统右键菜单项

例程“系统右键打开易语言.e”。运行程序添加菜单项，程序部分界面如图 10-13 所示。



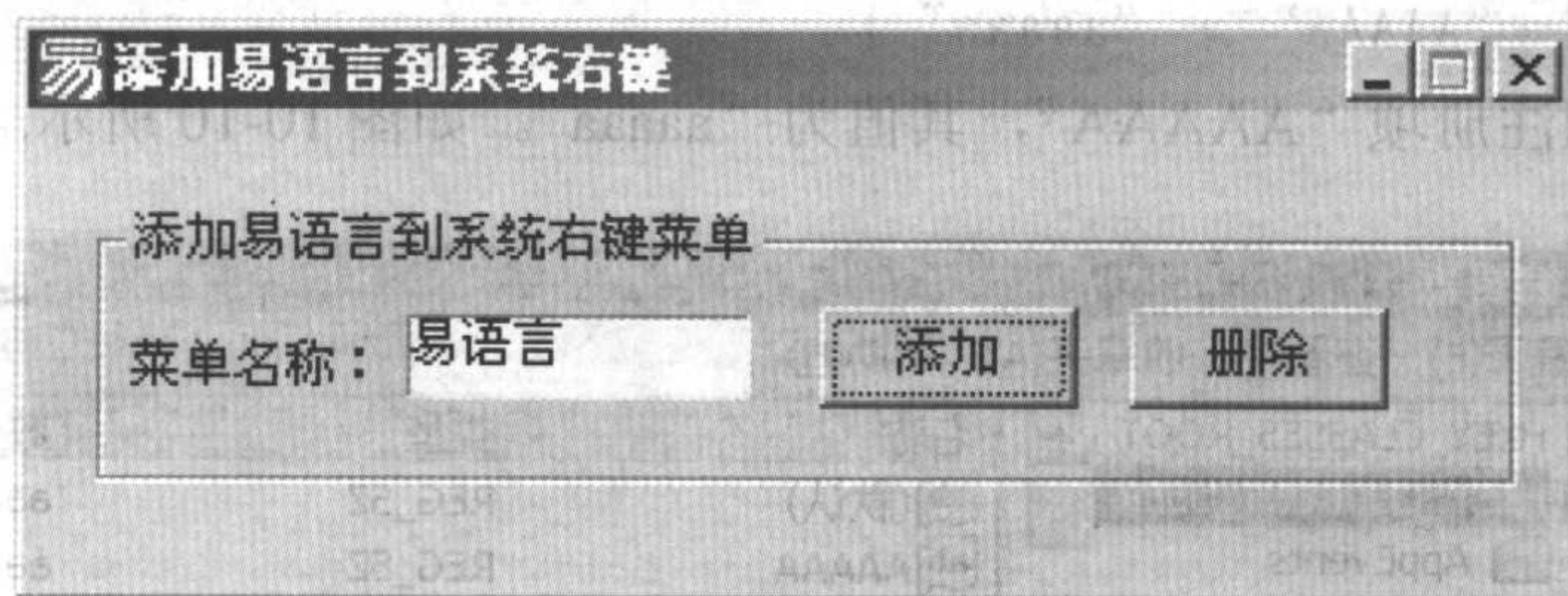


图 10-13 添加右键菜单项

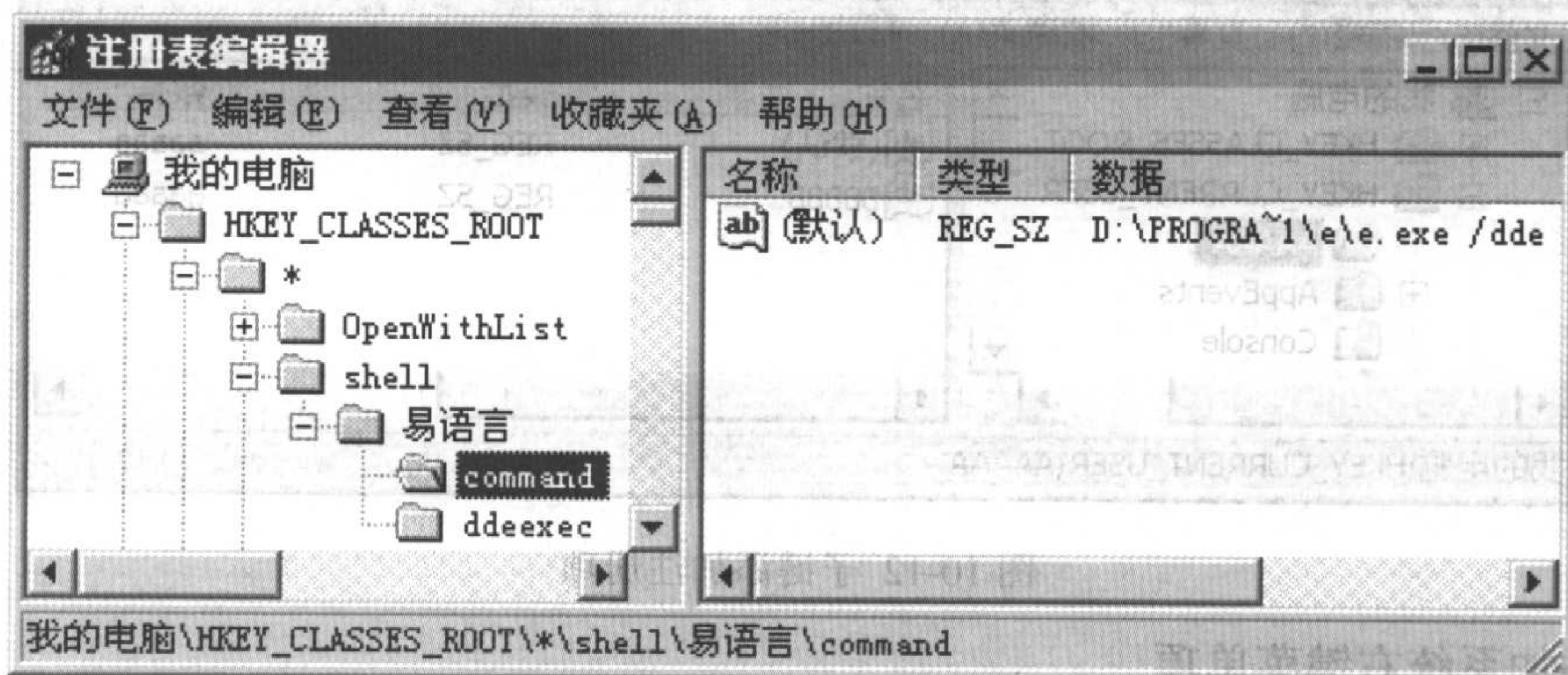
相关程序代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```
如果 (编辑框1.内容 ≠ "")
    如果 (寻找文本 (编辑框1.内容, "\", , 假) = -1)
        如果 (写注册项 (#根类, "*\shell\" + 编辑框1.内容 + "\command\"
            路径35 + ".exe /dde") = 真 且 写注册项 (#根类, "*\shell\"
            + 编辑框1.内容 + "\ddeexec\", "[open(\" + #引号 + \"%1\" +
            #引号 + \")])\" = 真)
            信息框 ("添加成功", 0, )
        信息框 ("添加失败", 0, )
    信息框 ("菜单名称中不能有" + #引号 + "\" + #引号, 0, )
信息框 ("菜单名称不能为空", 0, )
```

注意：上面程序中，为“ddeexec”项写入的数据中必须有“%1”。如果项目值中缺少“%1”，系统只会运行“易语言”，而不会打开选中的易源码文件。

注册表添加结果如图 10-14 所示。





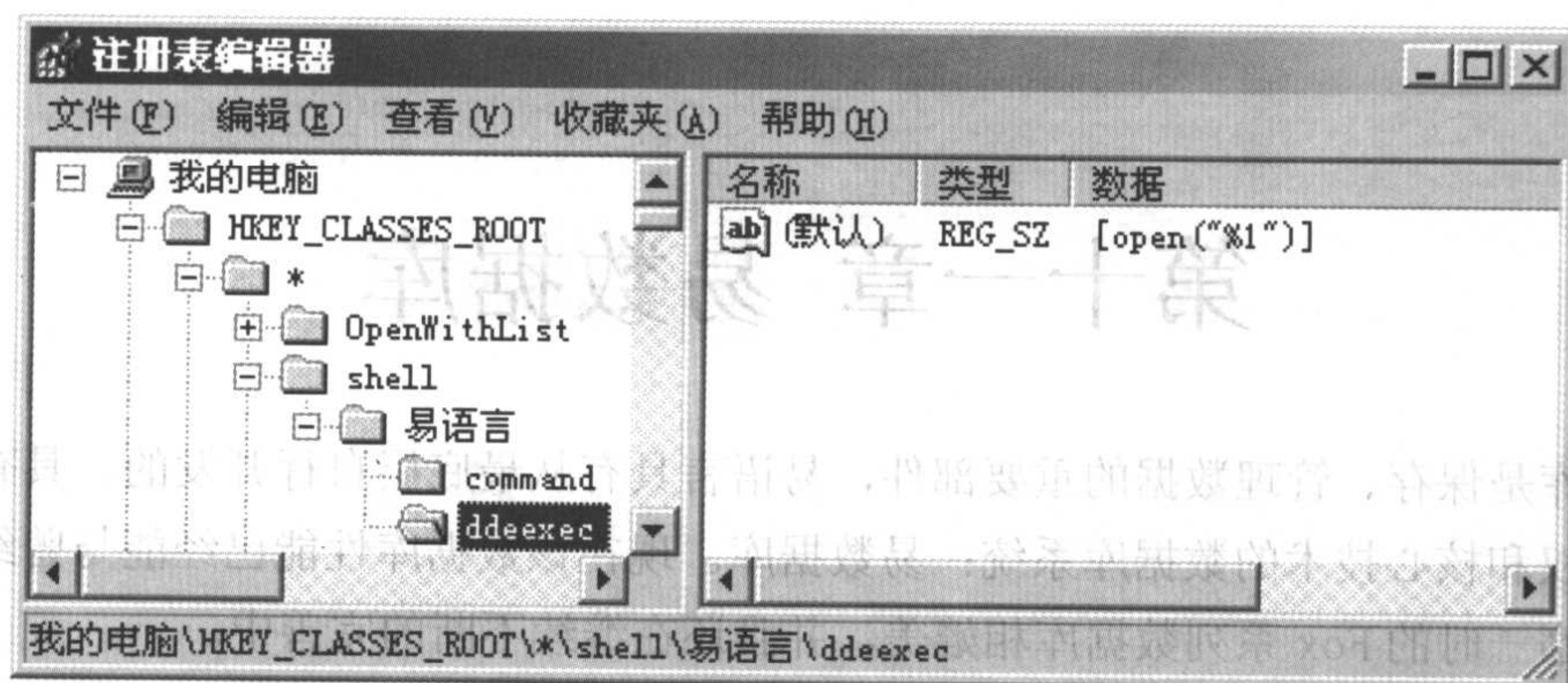


图 10-14 注册表项

上述程序运行后，当鼠标右键点击易源程序时，在系统菜单中就会有快捷方式。如图 10-15 所示。

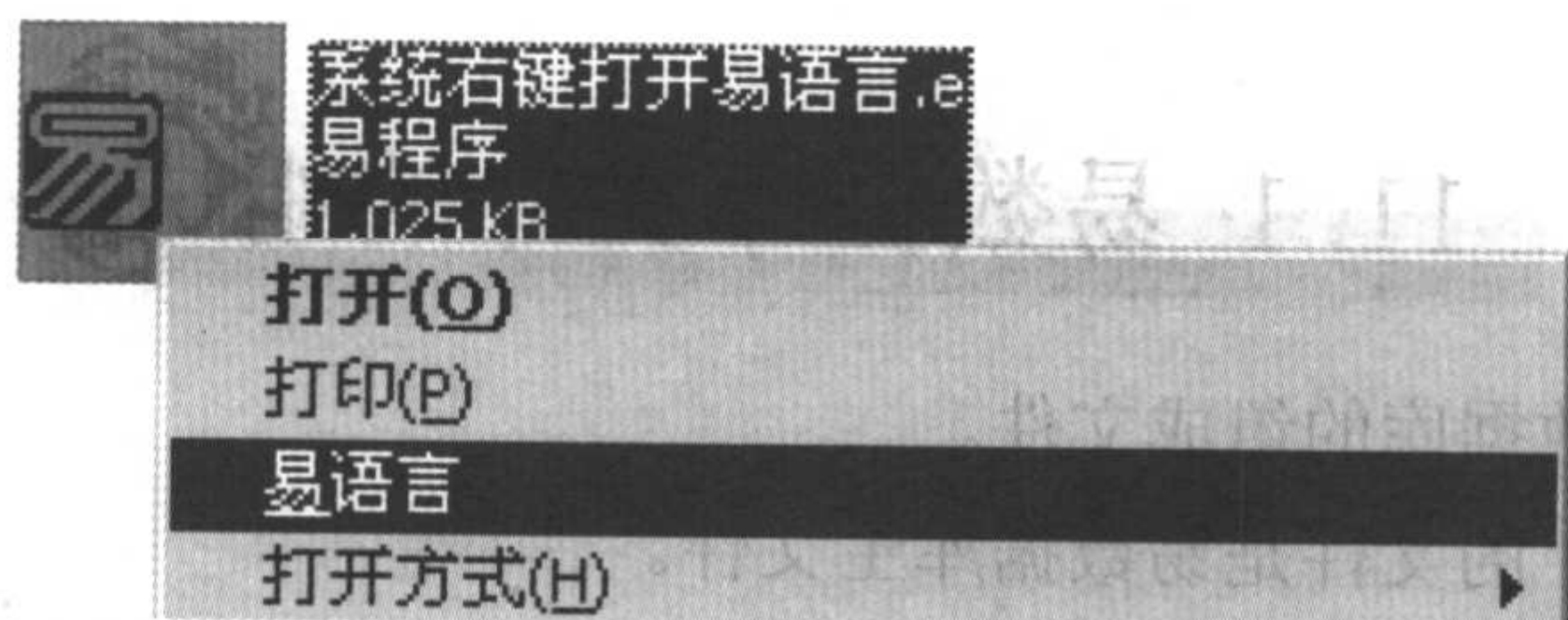


图 10-15 系统右键菜单

## 10.4 本章练习

编程序一定会接触到操作系统，对系统进行操作，有“运行( )”命令以运行外部程序，包括 IE 浏览器等，还有的是调用系统配置工具，在注册表的操作中，大家可以看出，如果熟练地操作注册表，还需要对注册表有非常深入的了解，这就需要大家进一步学习其他有关注册表操作的书籍了。

大家还可以进行以下的练习：

1. 制作一个带注册窗口的程序，显示当前电脑的硬件号，要求输入注册码，若注册码输入正确，就将注册码写到注册表中，以后运行时读取此注册码，如果正确就通过，否则就退出。另制作一个注册码生成器，可根据硬件号求出注册码，并提供给上述程序的使用者。
2. 编写一个易程序，具有中英文菜单的切换功能，切换的数据放在配置文件中，运行时可随时在两组菜单中切换。





## 第十一章 易数据库

数据库是保存、管理数据的重要部件，易语言具有从最底层自行开发的，具有完全自主知识产权和核心技术的数据库系统：易数据库。现在该数据库性能已经能与曾经在中国大地上风靡一时的 Fox 系列数据库相媲美，并且还在继续不断的完善中。

数据库可以方便地组织和管理数据。如制作一张工资表、建立通讯录、清查仓库记录、处理财务报表等，都要用到数据库操作。易数据库支持命令的具体解释请查看参考手册中相关内容。

### 11.1 易数据库文件的组成

先来介绍一下易数据库的组成文件。

扩展名为“.edb”的文件是易数据库主文件。

扩展名为“.edt”的文件是易数据库辅助数据文件，仅在数据库中存在备注型或者字节集型字段时才存在，文件名称除了后缀外与数据库主文件相同，它必须与.edb 文件放在同一目录中。

扩展名为“.enx”的文件是易数据库索引文件。索引文件由用户根据需要自行创建，使用它可加快查找记录的速度。

数据库由行和列组成，例如，一张工资表就是一个数据库中的二维表，如表 11-1 所示。

姓名	工资	扣除	实发
周丰	1500	200	
王自飞	2000	350	
张峰	1200	25	
李建业	1300	340	

表 11-1 工资表

其中每一行被称为一条记录，每一列被称为一个字段。表 11-1 的数据表就有四条记录和四个字段。

字段具有“名称”、“类型”、“最大文本长度”三个属性，相关介绍如下：

名称：字段名称的长度必须在 16 个字符以内。



类型：可以为以下常量值之一。#字节型；#短整数型；#整数型；#长整数型；#小数型；#双精度小数型；#逻辑型；#日期时间型；#文本型；#字节集型；#备注型。

最大文本长度：此属性仅当字段类型为“文本型”时才有效，用作指定文本的最大可能长度，其值范围必须在 1 到 1024 之间。如果字段类型不为“文本型”，本属性无效。当写入数据到数据库中的文本型字段内时，超出的部分将被自动剪裁。

## 11.2 使用工具创建和维护数据库

易语言内置了数据库维护的功能。使用菜单“数据库”→“结构编辑器”来创建一个指定结构的数据库或者修改一个现有数据库的结构。使用“记录编辑器”可以加入或修改初始记录数据，如图 11-1 所示。

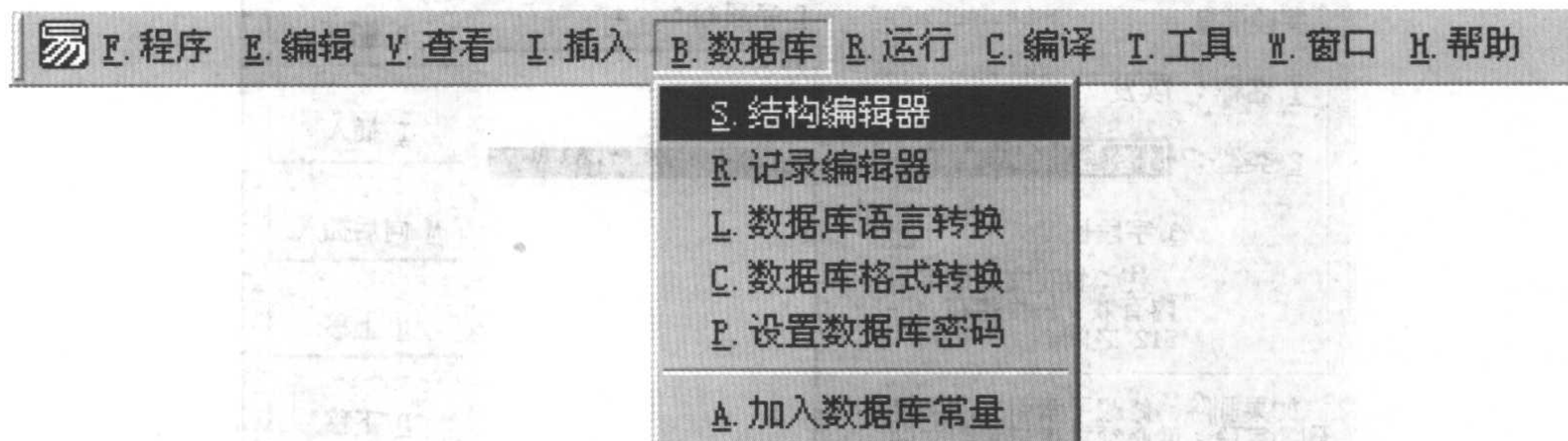


图 11-1 调用结构编辑器

值得注意的是，这两个选项的功能皆通过调用 dbmanger.exe 程序实现，该程序的源代码已经随系统一起提供（在“新建”窗口中的“打开例程”子夹中，选择“数据库管理器”），用户可以自行对其进行修改或者扩充，使之能满足用户的更多需要。

输入表 11-2 所示的名称和类型。

字段名称	数据类型	文本长度
姓名	文本型	8
工资	小数型	
扣除	小数型	
实发	小数型	

表 11-2 定义数据表

输入方法如图 11-2、图 11-3 所示。



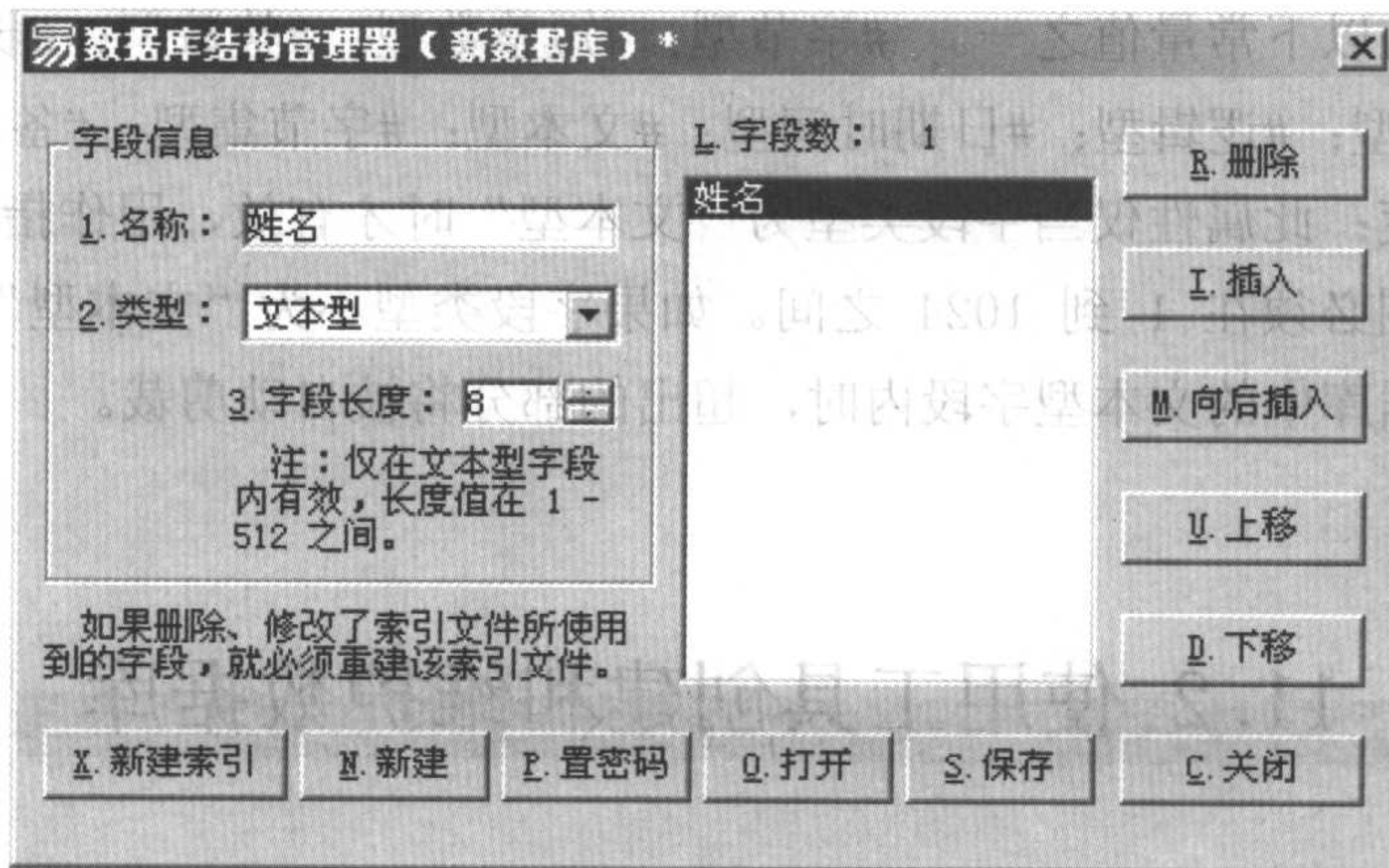


图 11-2 定义文本型字段

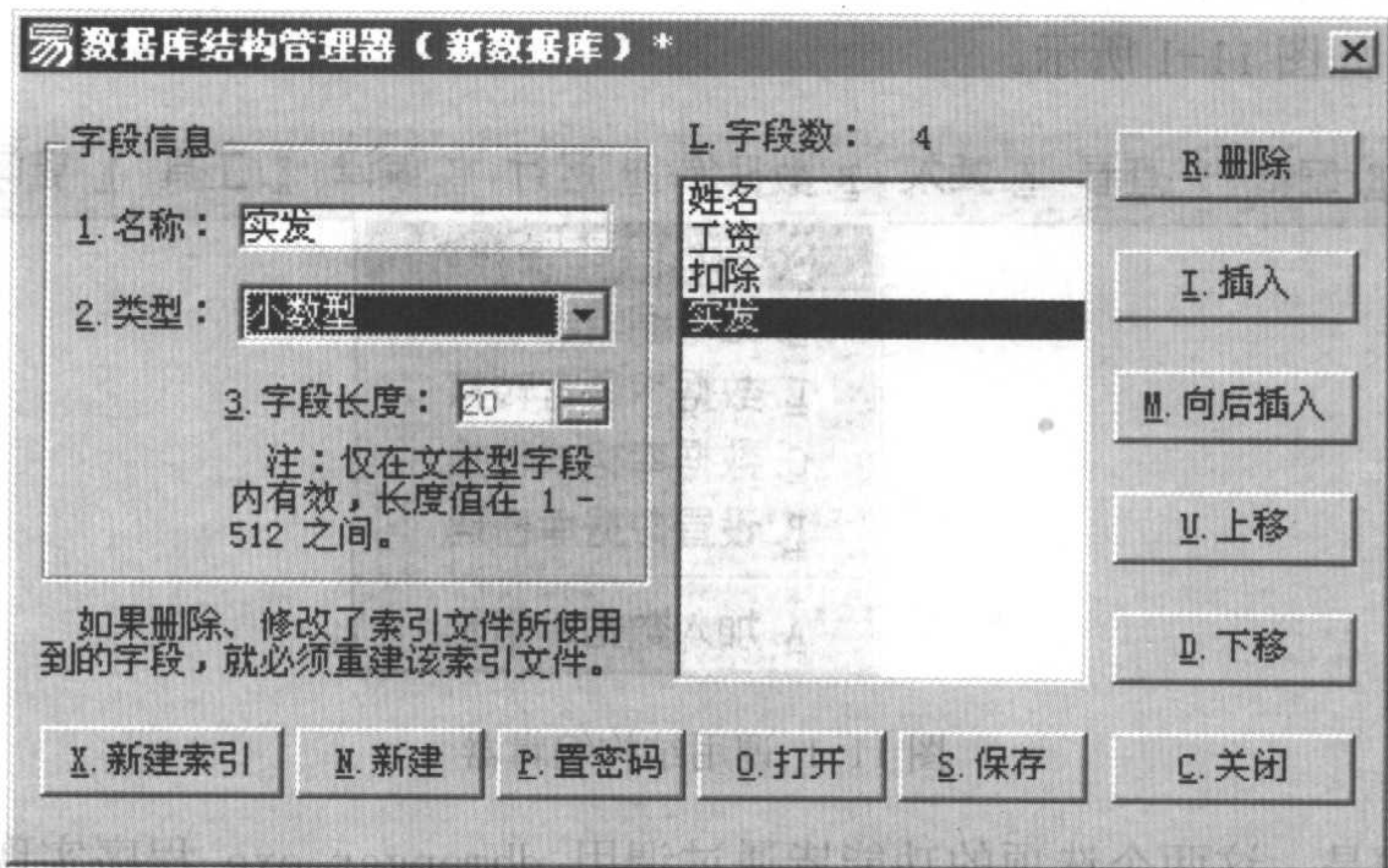


图 11-3 定义小数型字段

下面打开这个数据库进行记录操作，填充一些数据。使用菜单命令“数据库”→“记录编辑器”，打开“数据库记录编辑”对话框。如图 11-4 所示。

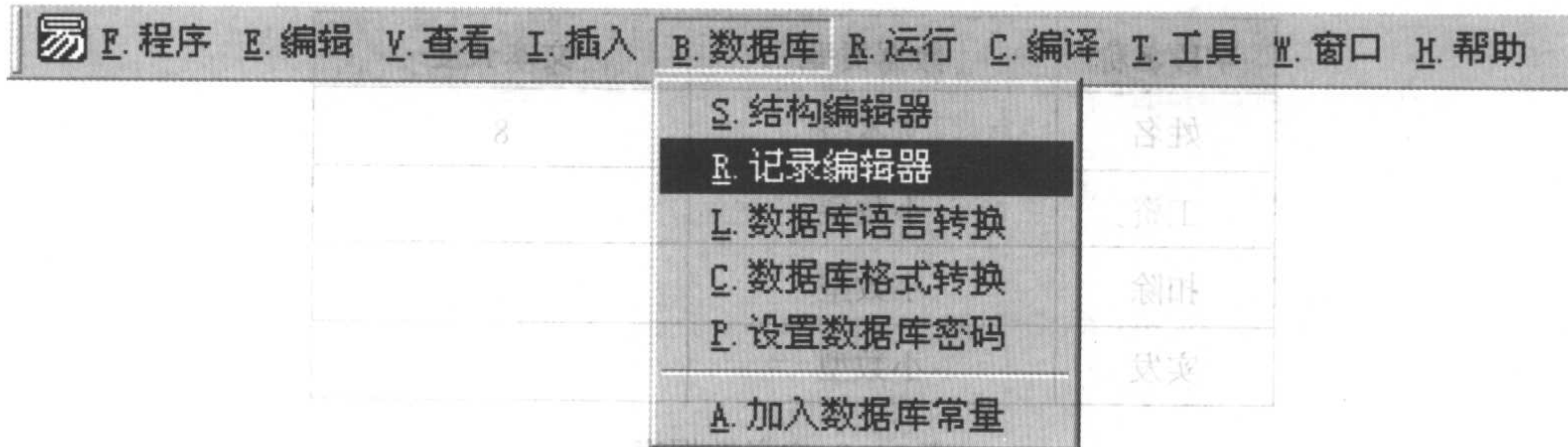


图 11-4 调用记录编辑器

打开刚才建立的数据库，这是一个只有字段而没有内容的空数据库，文件后缀为“\*.edb”。



可以看到，在这个程序中，可以进行录入，删除数据等操作。如图 11-5 所示。

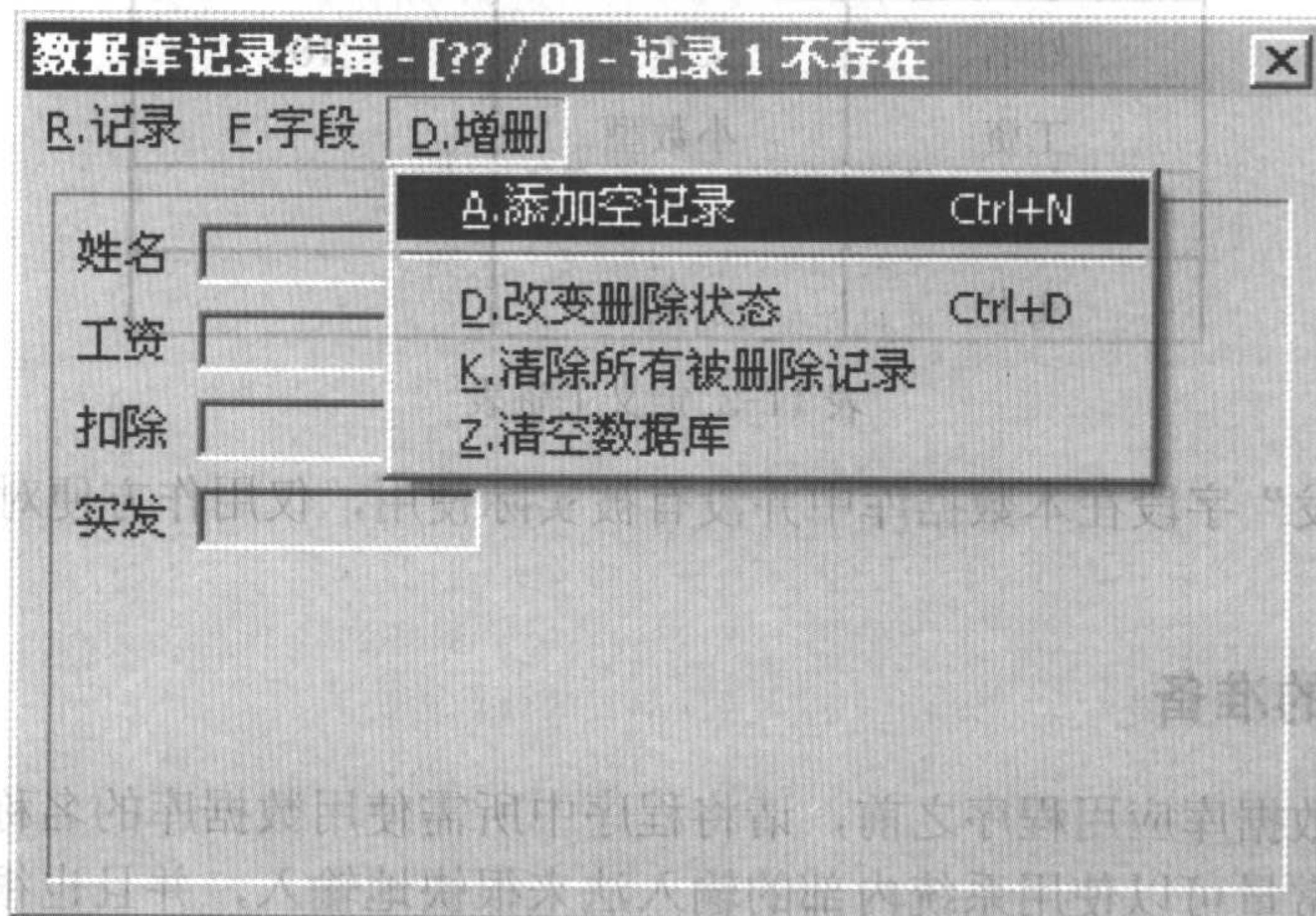


图 11-5 对数据记录进行操作

添加一条空记录后，就可以输入各字段数据了。如图 11-6 所示。

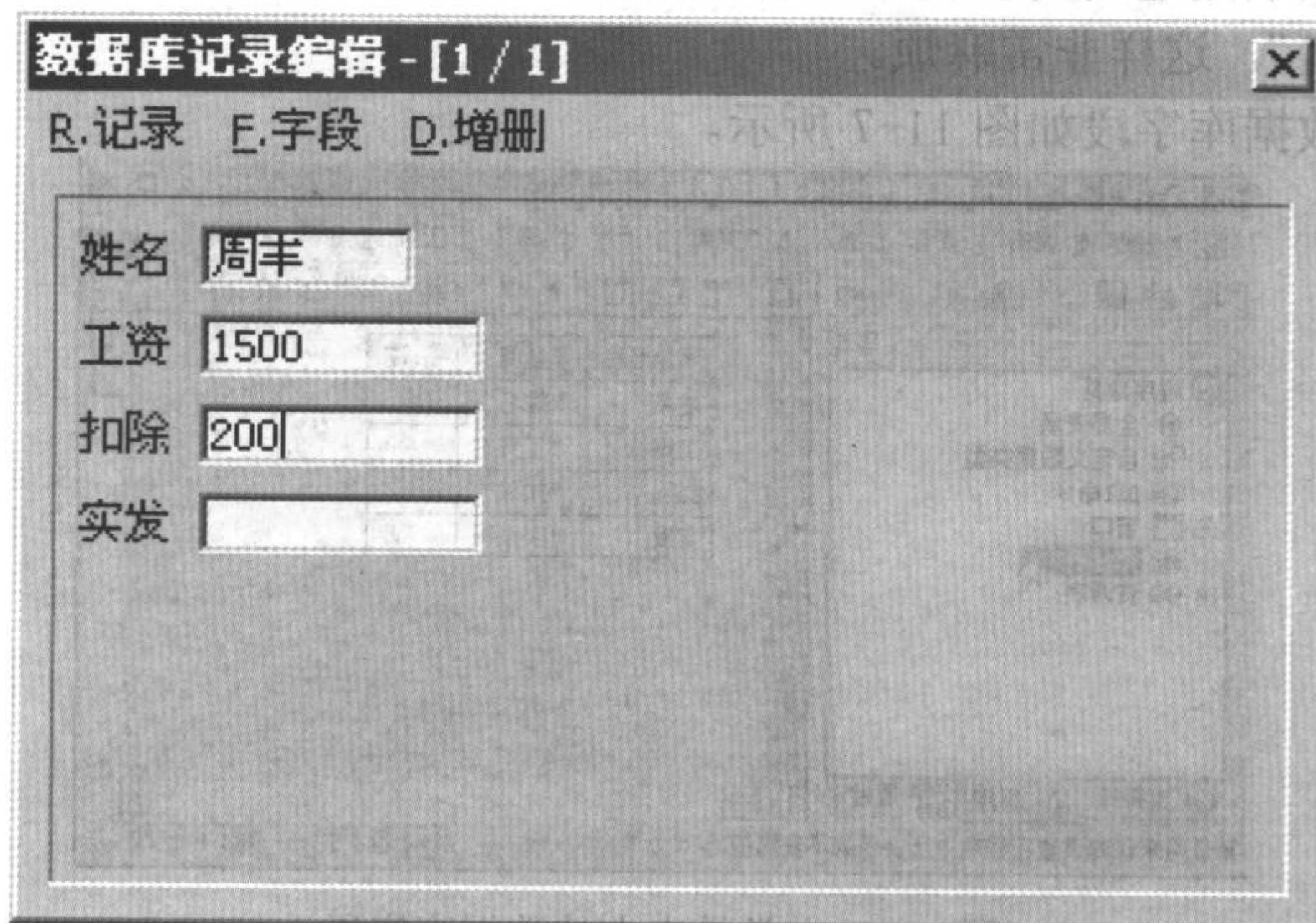


图 11-6 在空记录中输入数据

### 11.3 用命令创建数据库

上一节是使用工具进行数据库的创建和维护，但在实际工作中，对某些数据库结构无法预先知晓，如数据库格式转换时，不一定预先知道最终的数据库的结构，只能在程序中动态的生成数据库文件。本节介绍如何在易语言中，用编程的方法实现建立与维护数据库。

下面介绍一些数据库应用程序中常用的编程知识，首先请打开随书光盘本章目录中的“工资.edb”，其结构如表 11-3 所示。





字段名称	类型	最大文本长度
姓名	文本型	10
工资	小数型	
扣除	小数型	
实发	小数型	

表 11-3 定义工资表

其中的“实发”字段在本数据库中并没有被实际使用，仅用作方便对某些命令进行举例。

### 11.3.1 编程前的准备

在开始编写数据库应用程序之前，请将程序中所需使用数据库的名称及所有字段名设置为常量，因为常量可以使用系统内部的输入法来很快地输入，并且也很利于数据库的维护。

上面数据库内具有一个名称为“姓名”的字段，如果此字段名已经被设置为了常量，那么在程序中需要引用它时可以简单地输入#xm；如果没有设置为常量，就必须一个字一个字地输入“姓名”，这样非常麻烦。

常量表中的数据库字段如图 11-7 所示。

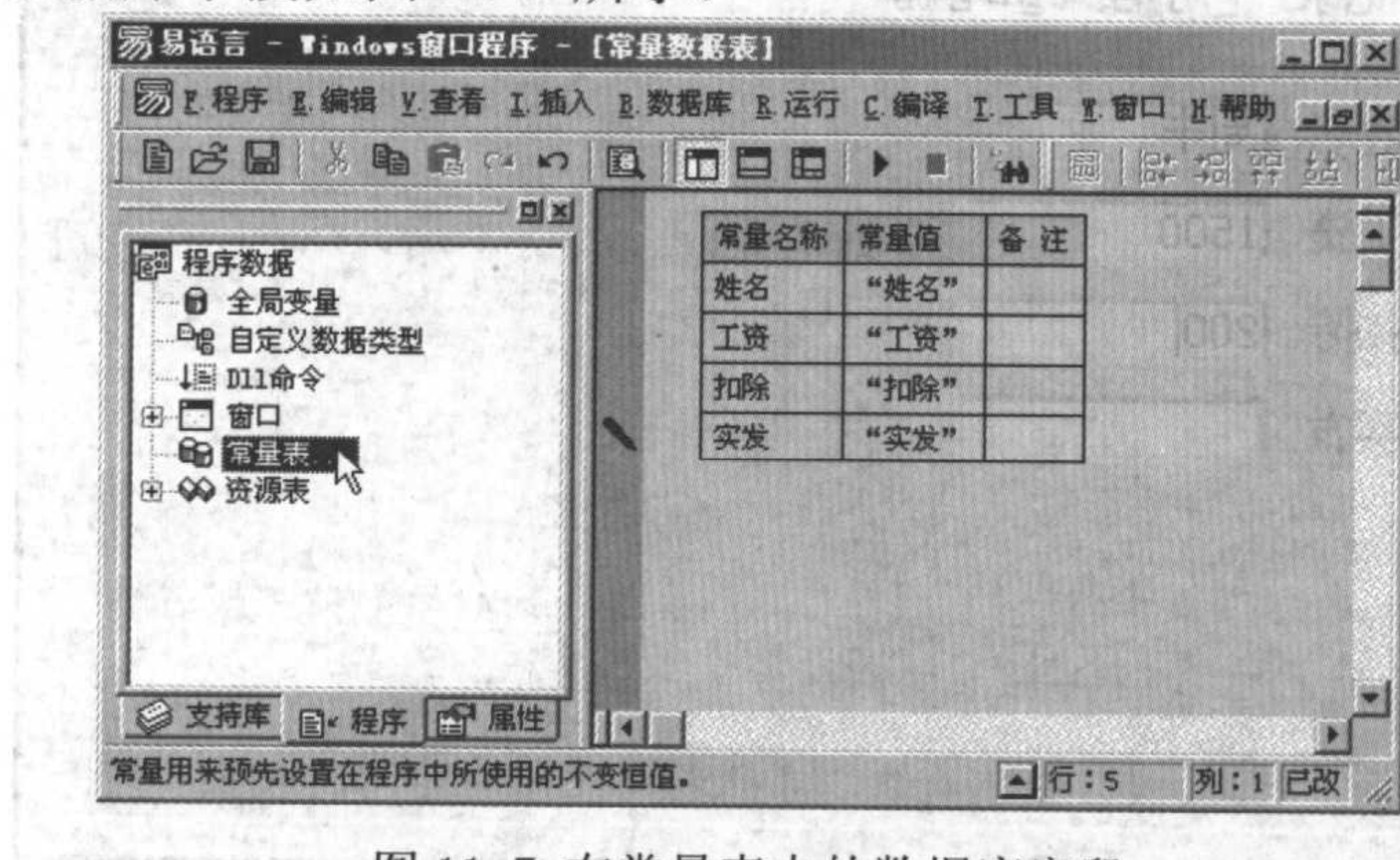


图 11-7 在常量表中的数据库字段

为了能够快速加入所有的相关数据库常量，在“数据库”菜单下有一个“加入数据库常量”功能，如图 11-8 所示。使用此功能可以一次性将指定数据库的名称及其所有字段名添加为常量。

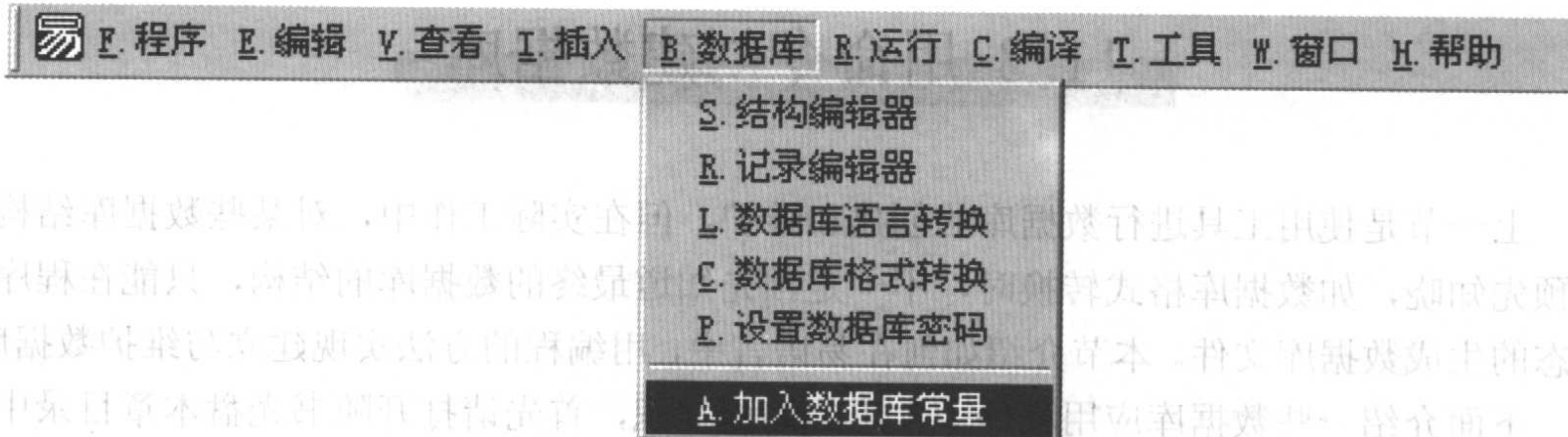


图 11-8 使用菜单加入数据库常量



### 11.3.2 创建数据库

在上一节中，建立了以常量作为字段名的数据库，在本节中，将用程序来实现这一任务。

首先新建一个易程序，加入两个“按钮”组件，“按钮 1”标题属性设置为“建立”，“按钮 2”标题属性设置为“打开”。

欲在程序中创建数据库请使用“创建( )”命令，如要创建上面的示例工资数据库“工资.edb”。在“\_按钮 1\_被单击”事件子程序中添加代码如下：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
字段表	字段信息		4	

字段表 [1]. 名称 = “姓名”

字段表 [1]. 类型 = #文本型

字段表 [1]. 最大文本长度 = 10

字段表 [2]. 名称 = “工资”

字段表 [3]. 名称 = “扣除”

字段表 [4]. 名称 = “实发”

连续赋值 (#小数型, 字段表 [2]. 类型, 字段表 [3]. 类型, 字段表 [4]. 类型)

创建 (“工资.edb”, 字段表)

通过上述程序，运行时单击“按钮 1”，就可以创建一个数据库文件。“创建( )”命令的第一个参数是欲创建数据库文件的路径，也可以是数据库的文件名，如“工资.edb”。如果只用数据库的文件名，那么此文件将被保存到当前易程序所在的目录中。

此数据库文件可以直接用“记录编辑器”打开，但没有记录显示，原因是数据文件中没有记录。此文件也可以用结构编辑器打开这个数据库进行查看。

### 11.3.3 打开数据库

欲打开指定数据库，请使用“打开( )”命令。可以同时打开多个数据库（易语言打开数据库的数量不受限制，其数目仅受 Windows 操作系统限制）。

准备用程序打开上节中的数据库，在“\_按钮 2\_被单击”事件子程序中添加代码如下：

子程序名	返回值类型	公开	备注
_按钮2_被单击			

如果 (打开 (“工资.EDB”, , , , , ) = 假)

信息框 (“无法打开工资数据库” + #左引号 + “工资.EDB” + #右引号 + “  
!”, #错误图标, “错误”)

结束 ()

信息框 (“打开” + #左引号 + “工资.EDB” + #右引号 + “数据库成功!”  
, #信息图标, “成功”)





运行程序，在单击打开数据库后，程序会显示打开成功的字样，如图 11-9 所示。

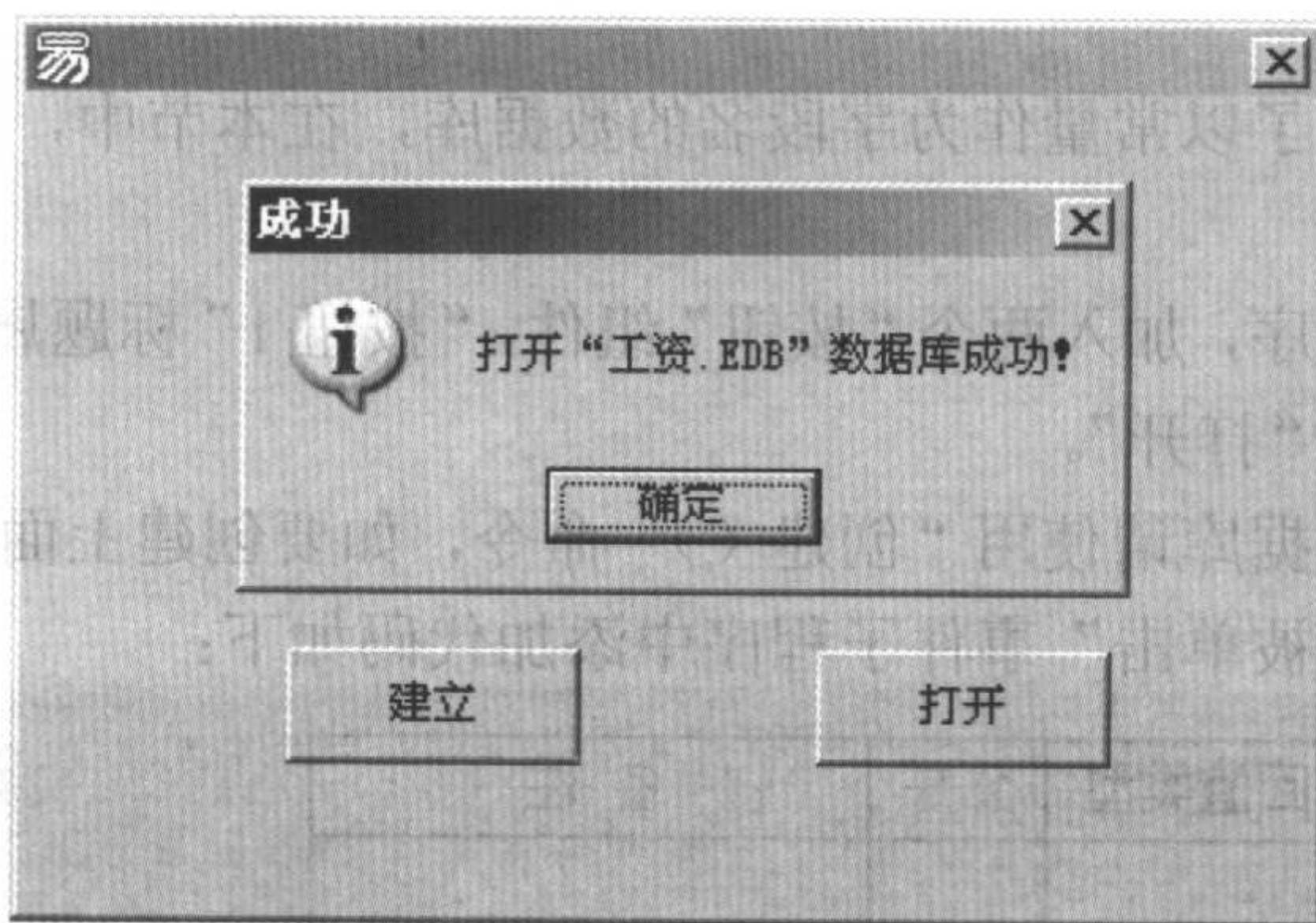


图 11-9 打开数据库成功

如果将上述 edb 数据库删除，再单击打开按钮，由于未找到“工资.edb”文件，所以打开数据库失败，如图 11-10 所示。

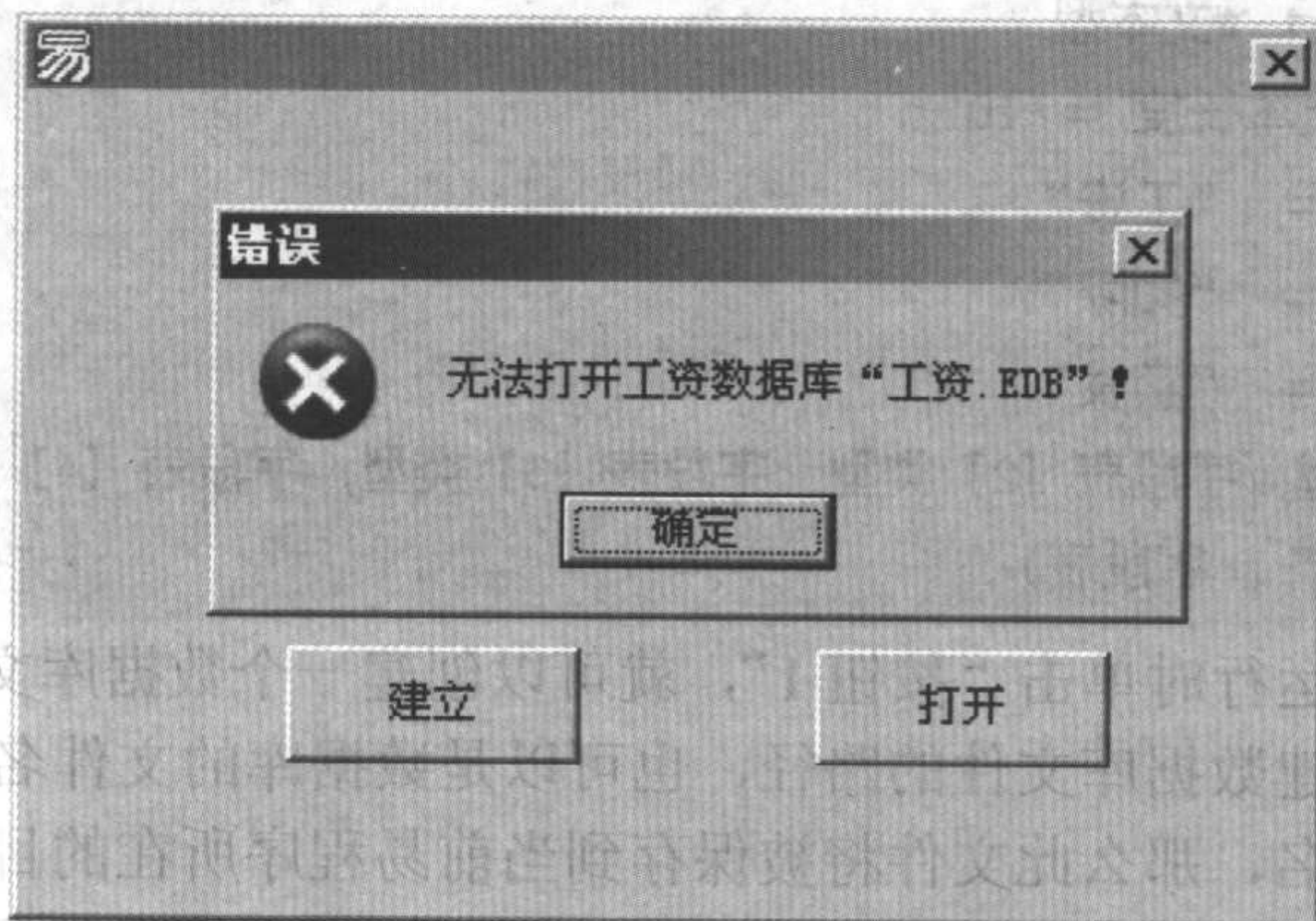


图 11-10 打开数据库失败

### 11.3.4 置当前数据库

系统内部有一个“当前数据库”状态值，它被用来指向某一个已被打开的数据库。绝大部分数据库操作命令都针对当前数据库，比如：记录指针的移动、字段的读写等等。用户可以使用“置当前库”命令来改变系统中“当前数据库”状态值的指向。实例代码如下：

```
打开（“工资”，，，，，，，）  
打开（“工资1”，，，，，，，）  
置当前库（“工资”）  
置当前库（“工资1”）
```

可以打开多个数据库，当增加记录时，就要将目标数据库置为当前库，以利于对记录进行操作。



### 11.3.5 关闭数据库

欲关闭当前数据库，请使用“关闭（）”命令。欲一次性关闭已打开的所有数据库，请使用“全部关闭（）”命令。实例代码如下：

关闭（）

全部关闭（）

## 11.4 用程序维护数据库

### 11.4.1 记录操作

易数据库中的命令都支持定义记录和字段范围，如“复制记录（）”命令能够将当前数据库的记录复制到另外一个数据库文件，其调用格式如下：

复制记录（数据库文件名，记录条件，字段范围，...）

其中，“记录条件”参数是用作让用户定义记录范围，用户需要使用一个子语句来提供参数数据。如将示例数据库中所有姓王的员工记录都拷贝到新数据库“工资2.edb”中，程序代码如下：

复制记录（“工资2”，读（#姓名）≈“王”，）

其中，读（#姓名）≈“王”即是用作定义记录范围的子语句。

**注意：**在易语言中“≈”符号使用“?=”输入。

假如你只想把示例数据库中的“姓名”和“工资”字段复制过去，就需要使用字段范围参数，程序代码如下：

复制记录（“工资2”，，#姓名，#工资）

字段范围参数为数据库命令的最后一个参数，以便让用户通过增加参数来同时提供多个字段，不过也可以使用数组来做同样的工作。具体操作为：建立一个局部变量，名称为“欲复制字段”，数据类型为“文本型”，数组成员数为0，程序代码为：

加入成员（欲复制字段，#姓名）

加入成员（欲复制字段，#工资）

复制记录（“工资2”，，欲复制字段）

假如想把数据库中所有员工的工资都加上100元，就需要使用数据库表达式。完成此工作的相关数据库命令“替换（）”的调用格式如下：

替换（替换范围，字段名称或替换数据，...）

每一个“字段名称或替换数据”参数对应“字段名称”、“替换数据”参数各一个。程序代码如下：

替换（真，#工资，读（#工资）+100）

其中，“#工资”为字段名称，“读（#工资）+100”即为数据库表达式，用作提供“工





资”字段替换数据的获取方法。

可以同时替换多个字段，如：“替换（真，#工资，读（#工资） + 100，#扣除，0）”语句可以同时将“扣除”字段清零。

数据库表达式中也可以同时使用多个字段，“替换（真，#实发，读（#工资） - 读（#扣除））”语句可以计算出每一个员工的实发工资。

还有很多数据库命令使用了数据库表达式，例如计算最大的实发工资并跳到其员工记录，代码如下：

```
取最大值（读（#工资） - 读（#扣除），）
```

根据实发工资排序到“工资2.edb”，代码如下：

```
计算排序（“工资2”，读（#工资） - 读（#扣除），，，）
```

## 11.4.2 当前记录指针

每一个被打开的数据库都有一个“当前记录指针”状态值，它指向数据库中的某一条记录，为一些记录读写命令提供位置指示，如读、写、删除等等。它除了指向正常的记录外，还可能具有以下状态值之一。

### 1. “首记录前（）”命令

表明当前记录指针已经移动到了数据库首记录的前面，此时如果执行读写当前记录的命令肯定会失败，因为无法找到对应的记录读写位置。使用“首记录前（）”命令可以测试到此状态值。

### 2. “尾记录后（）”命令

表明当前记录指针已经移动到了数据库最后一条记录的后面，此时如果执行读写当前记录的命令也会失败。使用“尾记录后（）”命令可以测试到此状态值。

使用“取记录号（）”命令可以取回当前记录指针所指向记录的编号（从1开始）。如果为0，表示在首记录前；如果大于最大记录编号，表明在尾记录后。

### 3. “到首记录（）”命令

该命令可以将当前记录指针移动到数据库的首记录上，“到尾记录（）”命令可以将当前记录指针移动到数据库的最后一条记录上，“跳过（）”命令可以相对移动当前记录指针值。

通过移动当前记录指针，可以遍历数据库中的所有记录，实例代码如下：

```
到首记录 0
```

```
判断循环首（尾记录后 0 = 假）
```

```
程序代码
```

```
跳过 0
```

```
判断循环尾 0
```

从数据库尾记录到首记录，反向遍历实例代码如下：



到尾记录 ()

--> 判断循环首 (首记录前 () = 假)

程序代码

跳过 (-1)

--- 判断循环尾 ()

### 11.4.3 读写字段

记录字段的读写均在当前数据库的当前记录处进行，主要为以下命令。

1. “读 ()”命令

如“读 (#姓名)”可以返回当前记录处员工的姓名。

2. “写 ()”命令

如“写 (#姓名, “张峰”)”可以将当前记录处员工的姓名改为“张峰”。

3. “读字段 ()”和“写字段 ()”命令

可以读写非当前数据库内的记录字段。

4. “修改 ()”命令

可以一次性修改当前记录的多个字段。例如：“修改 (“李铁”, 2000, 100)”语句可以将当前记录的员工姓名改变为“李铁”，工资改为 2000，扣除改为 100。

### 11.4.4 添加记录

1. 使用“加空记录 ()”命令可以在当前数据库的尾部添加一条新的空记录。

2. 使用“加记录 ()”命令可以同时提供欲添加的数据。例如“加记录 (“李铁”, 2000)”语句可以在当前数据库的尾部添加一条名为“李铁”，工资为 2000，扣除为 0 的新员工记录。

3. 使用“添加 ()”命令可以将其他数据库内的记录添加到本数据库，例如。

添加 (“工资 2”, , ) ' 可以将“工资 2.edb”数据库中的所有记录添加到当前数据库的尾部

添加 (“工资 2”, 取记录号 () ≤ 10, ) ' 语句仅可以添加前 10 条记录

添加 (“工资 2”, 是否已删除 () = 假, ) ' 语句仅可以添加所有未被删除的记录

### 11.4.5 删除记录

记录使用“删除 ()”命令删除后，并不马上从数据库中清除，仅被加上一个删除标记，记录依旧存在并可以正常访问。只有当执行“彻底删除 ()”命令后，这些被加上删除标记的记录才会从数据库中真正清除。

1. 使用“是否已删除 ()”命令可以查看当前记录是否被加上了删除标记，被加上删除标记的记录可以使用“恢复删除 ()”命令取消其删除标记。





2. 使用“清空( )”命令可以彻底删除当前数据库内的所有记录。

## 11.4.6 查找记录

查找记录有两种方法。不使用索引和使用索引。

### 1. 不使用索引

查找在示例数据库中所有姓“王”员工记录的方法。实例代码如下：

```
到首记录 0
--> 判断循环首 (查找 (#姓名) ≈ “王”)
    程序代码
    跳过 0
--- 判断循环尾 0
```

与易语言系统支持库中的拼音处理类命令结合，可以解决极常见的汉字近音搜寻问题。实例代码如下：

```
查找 (发音比较 (“张风”， 删全部空 (读 (#姓名)), 真))
查找 (输入字比较 (“zhangfeng”， 删全部空 (读 (#姓名)), 真, ))
查找 (输入字比较 (“ZF”， 删全部空 (读 (#姓名)), 真, ))
```

语句中的“删全部空( )”命令用作预先去除字段数据中可能存在的全半角空格，如果确定所有记录的该字段数据中都不存在空格，可以去掉此命令。

第一条语句可以查找出当前数据库中所有姓名发音为“张风”的员工，如：“张峰”、“张丰”等等；第二条语句完成类似的工作，不过使用的是全拼拼音编码；第三条语句使用的则是首拼拼音编码。

### 2. 使用索引

可以使用结构管理器预先在“工资.edb”上建立索引文件，也可以在程序中创建。实例代码如下：

```
新建索引 (“工资”， , , , #姓名)
```

上面的语句建立了一个名为“工资.idx”的索引文件，它基于示例数据库的“姓名”字段，也可以建立基于多个字段的索引文件。

索引只有被打开后才能被使用。使用“新建索引( )”命令新建的索引会被自动打开并设置为当前索引。对于已经存在的索引文件，必须在打开数据库时同步打开。

```
打开 (“工资”， , , , , “工资”)
```

此语句同步打开了“工资.idx”数据库索引文件，并将其设置为当前索引。

在数据库中搜寻所有姓名为“张峰”的员工。实例代码如下：

```
置当前索引 (“工资”)
到首记录 0
--> 判断循环首 (索引查找 (“张峰”))
    程序代码
    跳过 0
--- 判断循环尾 0
```



为了解决近音汉字搜寻问题,“易语言”中可以建立发音索引文件。实例代码如下:

新建索引 (“工资”, #读音索引, #不区分大小写 + #忽略所有空格, , #姓名)

新建索引 (“工资”, #南方读音索引, #不区分大小写 + #忽略所有空格, , #姓名)

以上两条语句分别建立一个基于标准读音和一个基于南方读音的索引文件。

使用发音索引文件就可进行近似音搜寻,如“索引查找 (“张风”)”就可以找到“张峰”。

在打开数据库时应该使用其“索引文件表”参数同时打开所有的索引文件,以便索引文件能够得到及时的更新。也可以使用“更新索引 ()”命令来强制更新当前索引文件。

大多数的数据库命令执行完毕后都会返回一个逻辑值,表明是否执行成功。但是也有一部分命令无法做到这一点,如“取最大值 ()”等。

大家可以在任何数据库命令执行完毕后立即调用“取错误码 ()”命令来查看其是否执行成功。如果成功,“取错误码 ()”命令将返回 0,否则将返回一个非 0 的错误值。另外,如果发现命令执行失败,立即调用“取错误信息 ()”命令可以取回对应的错误信息文本,当然是全中文的。

## 11.5 数据库实例

以上几节介绍命令较多,本节使用一个例子“好友通讯录.e”来具体指导大家如何使用数据库编程。

首先,新建一个易程序,自动生成“\_启动窗口”,在这个窗口上加入底图,添加两个按钮,一个是进入按钮,一个是退出按钮。主界面只是一个基本的框架,无需添加对数据库记录进行操作的组件。

再新建一个窗口,用于对记录进行操作。在程序面板中单击鼠标右键,即弹出含有插入新窗口的菜单,将窗口标题改为“好友录”。在窗口中放入标签组件,编辑框组件,一个图片框组件,一个通用对话框组件,一个组合框组件。窗口界面请参考例程。

返回到“\_启动窗口”,“\_按钮 1\_被单击”事件子程序中添加代码如下:

载入 (好友录, , 真)

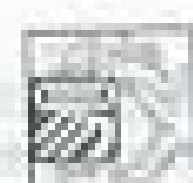
“\_按钮 2\_被单击”事件子程序代码如下:

结束 ()

在这个“\_启动窗口”中,还有一个非常重要的任务就是,启动时需要检查是否已经建立了数据库,如果没有建立,就要新建一个数据库。

在新建数据库之前,要将数据库的字段名放入常量表中。在程序面板中双击常量表,即可以在常量表中增加字段名称。如下所示。





常量名称	常量值	备注
朋友	“朋友”	
姓名	“姓名”	
网名	“网名”	
性别	“性别”	
邮编	“邮编”	
电话号码	“电话号码”	
手机号码	“手机号码”	
网络呼机	“网络呼机”	
传呼号码	“传呼号码”	
地址	“地址”	
电子信箱	“电子信箱”	
主页地址	“主页地址”	
备注	“备注”	
照片	“照片”	

“\_启动窗口\_创建完毕”事件子程序部分代码如下：

变量名	类型	静态	数组	备注
字段	字段信息		13	

```
--- 如果真 (文件是否存在 (“朋友.edb”) = 假)
    字段 [1].名称 = #姓名
    字段 [1].类型 = #文本型
    字段 [1].最大文本长度 = 10
    字段 [2].名称 = #网名
    字段 [2].类型 = #文本型
    字段 [2].最大文本长度 = 20
```

由于数据库为 13 个字段，必须为每个字段设置字段信息，此处略，完整代码参见例程。条件语句用来判断数据库文件是否存在。如果不存在，设置字段信息并创建新数据库；如果数据库文件存在就直接打开。代码如下：

```
--- 如果真 (创建 (#朋友, 字段) = 假)
    信息框 (“创建好友数据库” + #左引号 + “朋友.edb” + #右引号
        + “失败！”， #错误图标, “错误”)
    结束 ()

--- 如果真 (打开 (#朋友, , , , , ) = 假)
    信息框 (“无法打开好友数据库” + #左引号 + “朋友.edb” + #右引号
        + “！”， #错误图标, “错误”)
    结束 ()
```



通过以上的程序，就可以判断是否存在数据库，并且在不存在的情况下，可以立即生成空白数据库。

“\_好友录\_创建完毕”事件子程序。程序代码如下：

变量名	类型	静态	数组	备注
次数	整型			
变量	整型			

姓名框.获取焦点 ()

取记录数 ()

次数 = 取记录数 ()

到首记录 ()

--- 计次循环首 (次数, 变量)

    查询框.加入项目 (读 (1), )

    跳到 (变量 + 1)

    取记录号 ()

--- 计次循环尾 ()

这是为了将记录存入查询框，以及将光标移动至姓名框上。其中还用到一个“计次循环首 ()”命令。

在好友录窗口最下面的三个按钮是用来具体操作数据库记录的。其中标题为“添加好友”按钮的事件子程序是把窗口上与数据库字段相关联组件中的数据加入到数据库中，同时将组件中的数据清空。程序代码如下：

--- 如果真 (加入 () = 真)

    查询框.加入项目 (姓名框.内容, )

    连续赋值 (“”, 姓名框.内容, 网名框.内容, 邮编框.内容, 电话号码.内容, 手机号码.内容, 网络寻呼.内容, 传呼号码.内容, 地址.内容, 电子邮件.内容, 主页地址.内容, 备注.内容)

    性别框.现行选中项 = 1

    照片.图片 = { }

    姓名框.获取焦点 ()

其中“加入 ()”是一个子程序，作用是将组件中的数据加入到数据库中并要求用户必须为“姓名”字段提供数据。

第 2 行是将查询框中添加一个姓名作为快速查找。

后两行程序的作用是清空窗口中的编辑框，最后一行代码“姓名框.获取焦点 ()”将光标停留在姓名框上。

标题为“删除好友”按钮的程序代码中，第一个“如果真 ()”语句是判断数据库记录是否存在，第二个“如果真 ()”语句是判断是否真的要删除数据记录。如果条件成立，即进行删除的动作“删除 ()”；但此删除命令仅仅是作了一个删除的标记，没有真正的删除，因此还要有下面一行代码“彻底删除 ()”：将这条数据记录真正的删除掉。





最后3行程序代码的意义是：“到首记录（）”与“显示（）”命令将显示第一个记录，这样删除窗口中的信息后就不会为空了。“显示（）”是一个自定义子程序，显示数据库中的记录。最后一行程序代码表示同时将查询框中的这条记录也删除掉。

“显示（）”子程序代码中，第一个“如果真（）”判断语句是检查是否超出记录范围，如果超出，那么显示空的内容；如果没有超出，即将当前数据库中的记录显示在窗口中。

菜单的功能实现只要点取菜单中的项目，进入相应的子程序设计菜单，与按钮的内容一样即可。例如，添加好友菜单的程序内容，与添加好友的按钮内容一样。

## 11.6 数据库相关组件

程序运行过程中经常要用到大量的数据，但常规情况下数据（库）的显示与浏览是比较烦琐的（如果用标签、编辑框等通用组件来逐个显示数据，则往往需要很多组件，编程也十分烦琐）。为了简化易语言中的数据操作，易语言 2.0 版本之后提出了“数据应用框架”的概念。

“数据应用框架”最大的特点就是把数据、数据操作、数据显示分为三个不同的层次。每个层次由各自的组件完成相对独立的工作，至于各层次之间的联系，则由易语言在内部实现。这三个层次由低到高分别是：数据提供者、数据源、数据处理者。

其中，数据提供者用于存储、提供数据；数据源用于操作数据库；数据处理者用于显示数据。基本上数据提供者类似于商品仓库、后勤，数据源相当于包装车间或运输部门，而数据处理者相当于前台、展示柜台。

在易语言中，可充当数据提供者的组件有通用提供者、数据库提供者、外部数据库提供者等（外部数据库提供者组件将在下一章中介绍）；可充当数据源的只有数据源组件；可充当数据处理者的组件就很多了，最常用的是表格组件，此外还有编辑框、标签、列表框组件等，都可以作为数据处理者。见表 11-4 所示。

数据处理者	表格、编辑框、标签、图片框、组合框、列表框
数据源	数据源组件
数据提供者	通用提供者、数据库提供者、外部数据库提供者

表 11-4 数据应用框架分类

数据提供者、数据源、数据处理者三者之间必须事先“关联”起来，互相协调配合，才能共同完成对数据的处理。

“关联（）”的方法是：

1. 添加相应的组件，即数据处理者、数据源、数据提供者这三种类型的组件都必须存在。
2. 将数据源的“数据提供者”属性设置为某个数据提供者组件；
3. 将数据处理者组件的“数据源”属性设置为某个数据源组件。

举个例子说——比如要浏览/管理某个数据库，可以这样来编程序。



- (1) 首先在设计窗体上添加组件。数据库提供者，数据源，表格；
- (2) 设置数据库提供者组件的“数据库文件名”属性，即选择数据库文件 (\*.edb)；
- (3) 设置数据源组件的“数据提供者”属性为“数据库提供者 1”（从下拉列表中选择）。

(4) 设置表格组件的“数据源”属性为“数据源 1”（从下拉列表中选择）。

经过上述设置之后，大家可以发现，指定的数据库中的内容已经自动显示到表格组件中了！此后如果对数据源进行操作，数据库中的内容也会被同时改动，并且表格中的数据也会同时更新。

由于表格、数据源、通用提供者、数据库提供者的属性、方法、事件比较多，此处只介绍常用的几个。更详细的使用方法例程可参考易语言附带的“易之表.e”。

### 11.6.1 通用提供者、数据库提供者

通用提供者组件和数据库提供者组件都可充当“数据提供者”，属同一类组件，所以放在一起介绍。但它们也是有区别的。

1. 通用提供者。使用内存作为数据的存储仓库，全面支持所有数据操作接口。因此必要时可以将其他类型数据提供者内的数据导入到此类型中，全面发挥数据源对数据的操纵能力。

2. 数据库提供者。使用数据库作为数据的存储仓库，不支持以下数据操作接口：置行高、置类型、置文本色、置背景色、置字体名、置字体尺寸、置字体属性、置边距、置文本输入格式、置对齐方式、置密码方式、合并、分解、加线条、删线条、初始尺寸时同时改变列数、在中间插入行、插入列、删除列。

如果想对数据库提供者中的数据进行以上操作，应该先将数据通过数据源导出到通用提供者中。

#### 3. 通用提供者的重要属性

“初始行数”、“初始列数”属性均为整数型。指定初始数据的行、列数。默认值都是 0。

当通用提供者跟数据源、表格正确关联后，如果不设置这两个属性，表格中仍然一片空白，看不出一点表格的样子（因为初始行列数默认值都是 0）。为了美观可自由设定某个值。

#### 4. 数据库提供者的重要属性

##### (1) “数据库文件名”属性

文本型，指定欲操作的数据库全路径文件名 (\*.edb)。

这是数据库提供者最重要的一个属性。

##### (2) “字节集字段处理”属性

整数型，指定对字节集类型字段的处理方式。有以下可选值：0. 跳过；1. 视为图片数据；2. 视为字节集数据，默认值是 0。

如果确定数据库中没有字节集类型字段，可设置为 0；如果确定数据库中字节集字段为图片数据，则设置为 1，否则设置为 2。





**注意：**对“数据提供者”组件中的数据进行处理，是通过调用“数据源组件”的“方法”来实现的。通过把数据源组件和数据提供者组件“关联”起来，数据源就可以处理数据提供者组件中的数据了。（关于“关联”的方法本章一开始就已经介绍过了）就是把数据提供者属性设置为相应的数据提供者组件，用代码可表示为：

数据源 1. 数据提供者 = “数据库提供者 1”

或

数据源 1. 数据提供者 = “通用提供者 1”

### (3) “数据库密码”属性

为保护数据库的安全，需要为数据库添加密码。选择菜单“数据库”→“结构编辑器”，或使用“置数据库密码（）”命令为一个打开的数据库更新密码。

当数据库有密码时，可以通过在本属性中填写密码来顺利的打开数据库。

### 5. 具体应用

#### (1) 将数据库提供者中的数据“导入”到通用数据提供者的方法

将数据库提供者中的数据“导入”到通用数据提供者中，方法其实很简单，只需用数据源组件的“添加（）”方法就可以实现。

数据源 1. 添加（数据库提供者 1，，）

或

数据源 1. 添加（数据源 2，，）

打开书中例程“导入.e”。标题为“显示数据”的按钮演示“导入”功能。相关程序代码如下：

通用对话框1. 初始目录 = 取当前目录 0

--- 如果真（通用对话框1. 打开（） = 真）

数据库提供者1. 数据库文件名 = 通用对话框1. 文件名

数据源1. 初始尺寸（0，0）

数据源1. 添加（数据库提供者1，，）

#### (2) 选用原则

作为最重要的两个数据提供者，通用提供者组件和数据库提供者组件的关系是：前者功能强大，可对数据进行各种操作，但没有直接的数据来源；后者功能受限，只能完成对数据的基本操作，但可以直接连接到数据库。二者的选用原则是：

- 如果要使用数据库，且对数据的操作不涉及外观设置（如修改字体、颜色或单元格线条等），可选用数据库提供者。
- 如果需要使用数据库，又想对数据进行外观设置，可同时使用数据库提供者和通用提供者组件（先把数据库提供者中的数据“导入”通用提供者中，再对后者操作）。
- 如果没有用到数据库，可单独选用通用提供者。

## 11.6.2 数据源

数据源是数据提供者与数据处理者之间的纽带。



### 1. 数据源的重要属性

#### (1) “数据提供者”属性

文本型，指定本数据源所基于的数据提供者组件名。

本属性的设置方法与表格组件的“数据源”属性的设置方法类似，即：在属性面板中设置本属性时，如果此时窗体上已经放置了某个（或多个）数据提供者组件，则会以下拉列表的形式列出，选择其中之一即可。

如果要在程序中用代码的方式为本属性赋值，需以文本形式指明数据提供者组件名称。实例代码如下：

```
数据源 1. 数据提供者 = “通用数据提供者 1”
```

#### (2) “只读”、“允许添加”、“允许删除”属性

这三个属性都是逻辑型，用于限制用户对数据操作的权限。

**提示：**因为数据源组件上的添加删除记录的按钮不利于数据的完全操作，实际应用中，通常把数据源组件的“可视”属性设置为“假”，使它对用户不可见，令用户无法直接操作数据。

确实需要添加或删除记录时怎么办呢？可以调用数据源组件的如下方法：插入行，添加行，删除行，插入列，删除列等。

### 2. 数据源的重要事件

数据源组件的重要事件有：“当前记录被改变”，“添加记录”，“删除记录”。

#### (1) “当前记录被改变”事件

事件的产生时机分别为：当操作者按下数据源的前四个按钮时产生本事件。

若数据源组件的“可视”属性被置为“假”时，用户是不会看到该组件的，所以将不会触发该事件。

#### (2) “添加记录”事件

当按下添加记录按钮添加了新记录时，产生本事件。

#### (3) “删除记录”事件

按下删除记录按钮删除了当前记录时，产生本事件。

打开随书例程：“导入.e”，点击数据源的六个按钮，看看会发生什么事情。

### 3. 数据源的专有方法

数据源组件的方法非常多，请大家查看支持库面板中的“数据类型”→“数据源”，提示面板里面将列出数据源组件的所有属性、方法和事件（显示于状态夹中）。

为了便于理解，下面将数据源组件的方法稍加分类：

(1) 对记录的操作类方法：“到首记录()”，“到尾记录()”，“跳过()”，“跳到()”，“取记录号()”；

(2) 对行列的操作类方法：“取行数()”，“取列数()”，“插入行()”，“添加行()”，“删除行()”，“插入列()”，“删除列()”；

(3) 数据存取操作类方法：“置文本()”，“取文本()”，“置数据()”，“取数据()”，“添加()”，“初始尺寸()”，“存到字节集()”，“从字节集读()”，“存到文件()”，“从文件读()”，“单元格到字节集()”，“字节集到单元格()”，“单元格到文件()”，“文件到单





元格()”，“刷新()”，“保存更改()”；

(4) 对外观的操作类方法：“置表头行数()”，“置表头列数()”，“置行高()”，“置列宽()”，“置文本色()”，“置背景色()”，“置字体名()”，“置字体尺寸()”，“置字体属性()”，“置边距()”，“置对齐方式()”，“置初始属性()”；

(5) 对单元格操作类方法：“合并()”，“分解()”，“是否被合并()”，“加线条()”，“删线条()”，“是否有线条()”，“清除()”；

(6) 关于打印操作类方法：“打印设置()”，“置打印设置()”，“取打印设置()”，“取打印页宽()”，“取打印页高()”；

易语言附带的例程“易之表.e”中，演示了数据源组件及表格组件的绝大多数属性方法，请自行参考。

### 3. 用按钮模拟数据源组件

这里的模拟并不是不用数据源组件，而是将数据源组件“可视”属性设置为“假”。打开书中例程。“数据源-模拟按钮.e”。如图 11-11 所示。

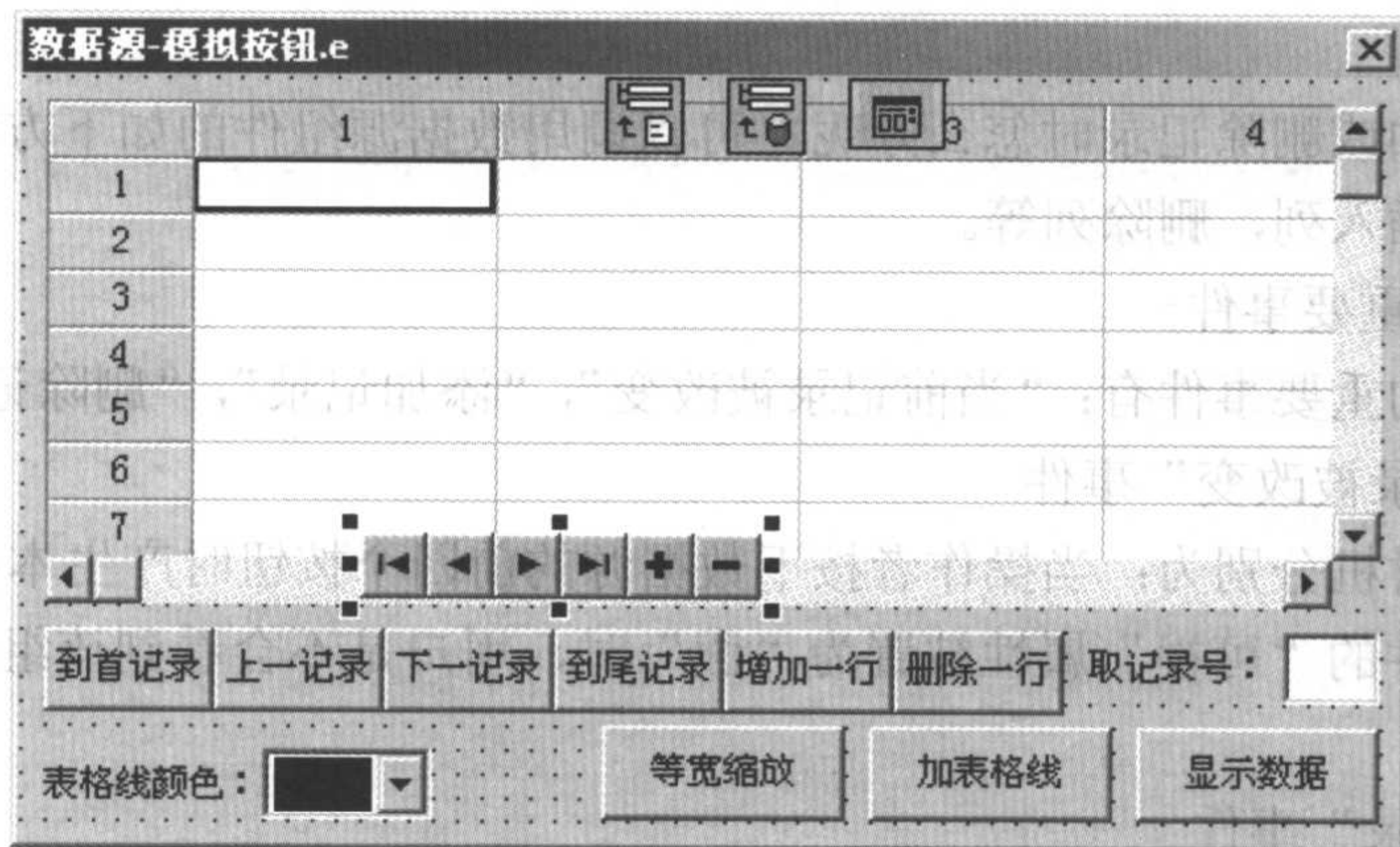


图 11-11 “数据源-模拟按钮.e”程序设计界面

标题为“到首记录”按钮事件子程序实例代码如下：

数据源 1. 到首记录 ()

标题为“到尾记录”按钮事件子程序实例代码如下：

数据源 1. 到尾记录 ()

可以看出上述程序代码是使用了数据源的两个方法命令。

标题为“上一记录”按钮事件子程序实例代码如下：

数据源 1. 跳到 (表格 1. 取光标行号 () - 1)

标题为“下一记录”按钮事件子程序实例代码如下：

数据源 1. 跳到 (表格 1. 取光标行号 () + 1)

可以看到，这次没有用数据源组件的命令取行号，而是使用了表格组件取行号，这是因为表格如果添加了空白行的话，它的行号比数据源中的记录号要多，因此取表格的光标行号要更加准确一些。由于表格显示的是数据源保存的数据，而数据源保存的数据是由数据库提供者提供的，所以直接对表格中数据的操作也就是间接操作数据库文件，但这需要数据库命令和“数据源”组件方法的配合使用。



增加行程序代码如下：

```

-- 如果 (数据源1.数据提供者 ≠ “数据库提供者1”)
-- 数据源1.插入行 (表格1.取光标行号 0, )
-- 数据源1.插入行 (数据源1.取行数 0 + 1, )
-- 表格1.置光标 (数据源1.取行数 0, 1)

```

删除行程序代码如下：

```

数据源1.删除行 (表格1.取光标行号 0, 表格1.取选择行数 0)

```

可以看到“增加一行”的按钮使用了取数据源中的所有记录行数，即为数据源的末尾加空白行。而“删除一行”的按钮使用了取表格光标行号的命令，这样光标所在行就会被删除。

为什么增加一行对于数据源为“数据库提供者”来说只能在末尾增加，而不能在当前选中行后面增加，而当数据源为“通用提供者”时可以在任意地方增加呢？这就是本书前面提到的，通用提供者要比数据库提供者灵活，所以有的功能对于数据库提供者来说用不上。

#### 4. 易之表 grd 文件与数据库 edb 文件

可以将存在于通用提供者中的数据内容保存为 grd 文件，grd 文件在易语言中也称之为易之表文件，这是因为易语言提供了一个工具“易之表.e”可由它保存生成的 grd 文件。

存在于数据库提供者中的数据库 edb 文件，只要是数据库作为通用提供者，那在数据源及表格中的修改可以立即生效并存回数据库文件中。

打开书中例程“易之表.e”，并运行它，可以在开始菜单中看到一个是“打开”菜单，另一个是“修改数据库”。这两者有什么区别吗？为什么不用一个命令打开呢？

使用“打开”菜单打开的是 grd 文件，使用“通用提供者”组件提供的“数据源”。

而用“修改数据库”菜单打开的是 edb 数据库文件，使用的是“数据库提供者”作为“数据源”，因此两个菜单实现的功能并不一样。

本节一开始就提到通用提供者比数据库提供者灵活，在这里，可以用两个不同的菜单体会两者的区别。

用打开命令打开 grd 文件，若没有找到 grd 也可以在易之表初次运行时任意填充一些数据在表格中，因为启动时程序默认为“通用提供者”提供数据源。在这里将鼠标放在两行表头前，可以调整表格的行宽和列高。

保存 grd 文件使用的是“存到文件 ( )”方法命令。

“易之表.e”中，保存菜单使用的“保存文档”事件子程序中相关程序代码如下：

```

数据源 1. 存到文件 (保存文件名)

```

打开 grd 文件使用的是“从文件读 ( )”方法命令。

“易之表.e”中，打开菜单使用的“打开并读入文件”事件子程序，相关程序代码如下：

```

数据源 1. 从文件读 (文件名)

```

#### 5. 数据源的方法





### (1) “初始尺寸( )”方法

通用提供者中也有两个“初始行数”、“初始列数”属性，与这个方法类似。

初始数据源中数据的行列数，数据源中全部原有数据将被清除。注意某些数据提供者可能不支持此方法。成功返回真，失败返回假。“初始尺寸”带有两个参数。

参数<1>的名称为“行数”，类型为“整数型(int)”，可以被省略。如果本参数被省略，默认值为1。

参数<2>的名称为“列数”，类型为“整数型(int)”，可以被省略。如果本参数被省略，默认值为1。

“易之表.e”中在“新建”菜单中的程序代码如下：

数据源 1. 初始尺寸 (50, 10)

这就表示在新建一个新的 grd 文件时将空表的行数定为 50 行，空表的列数定为 10 列。

下面的方法全部都是以“易之表.e”为例学习的，为了能快速找到大家想要的命令，可以使用易语言的快速查找功能。

打开例程“易之表.e”，双击窗体，进入程序设计界面，使用 Ctrl+F 组合键即可以在程序设计界面中查找，输入你想要找的命令，如输入“初始尺寸”后回车就可以快速查找到，如果想继续查找，可以使用 F3 热键。

### (2) “打印设置( )”方法

调用格式：〈逻辑型〉 对象. 打印设置 ( )

调用对话框设置数据的打印配置信息。当操作者按确认按钮退出对话框时返回“真”，否则返回“假”。

在“打印”菜单中的程序代码如下：

```
--- 如果真 (数据源1. 打印设置 ( ))
    设置修改 ( )
    表格1. 打印 ( )
```

### (3) “清除( )”方法

调用格式：〈无返回值〉 对象. 清除 (行号, 列号, [行数], [列数])

清除数据源中指定范围内单元格内容为空文本。

在“剪切”菜单中的程序代码如下：

数据源 1. 清除 (表格 1. 取光标行号 ( ), 表格 1. 取光标列号 ( ), 表格 1. 取选择行数 ( ), 表格 1. 取选择列数 ( ))

### (4) “合并( )”方法

调用格式：〈无返回值〉 对象. 合并 (行号, 列号, [行数], [列数])

组合数据源中指定范围内的单元格，使之以一个单元格的形式表现。**注意：**如果数据源所使用的数据提供者不支持此特性，本命令将被忽略。

在“组合”菜单中的实例代码如下：

设置修改 ( )

数据源 1. 合并 (表格 1. 取光标行号 ( ), 表格 1. 取光标列号 ( ), 表格 1. 取选择行



数 ( ), 表格 1. 取选择列数 ( )

(5) “分解 ( )” 方法

调用格式: 〈无返回值〉 对象. 分解 (行号, 列号)

分解数据源中指定的已经组合的单元格, 行列参数指向被组合单元格内的任何一个单元格即可。注意如果数据源所使用的数据提供者不支持此特性, 本命令将被忽略。本命令为初级对象成员命令。

在“分解”菜单中的程序代码如下:

设置修改 ( )

数据源 1. 分解 (表格 1. 取光标行号 ( ), 表格 1. 取光标列号 ( ))

(6) “插入列 ( )” 方法

调用格式: 〈逻辑型〉 对象. 插入列 (列号, [列数])

在数据源中指定位置处插入新数据列。成功返回真, 失败返回假。

在“插入列”菜单中的程序代码如下:

数据源 1. 插入列 (表格 1. 取光标列号 ( ), )

(7) “删除列 ( )” 方法

调用格式: 〈逻辑型〉 对象. 删除列 (列号, [列数])

在数据源中指定位置处删除数据列。成功返回真, 失败返回假。

在“删除列”菜单中的程序代码如下:

数据源 1. 删除列 (表格 1. 取光标列号 ( ), 表格 1. 取选择列数 ( ))

(8) “置行高 ( )” 方法

调用格式: 〈无返回值〉 对象. 置行高 (行号, [行数], 高度)

设置数据源中数据行在表现时的高度, 单位为 0.1 毫米。注意如果数据源所使用的数据提供者不支持此特性, 本命令将被忽略。

在“置行高”菜单中的程序代码如下:

变量名	类型	静态	数组	备注
行高	整数型			

如果真 (输入框 (“请输入新行高 (单位 0.1 毫米)”, “新行高”,  
行高, #输入整数) 且 行高 > 0)

数据源 1. 置行高 (表格 1. 取光标行号 0, 表格 1. 取选择行数 0, 行高)

设置修改 0

(9) “置列宽 ( )” 方法

调用格式: 〈无返回值〉 对象. 置列宽 (列号, [列数], 宽度)

设置数据源中数据列在表现时的宽度, 单位为 0.1 毫米。注意如果数据源所使用的数据提供者不支持此特性, 本命令将被忽略。

在“置列宽”菜单中的程序代码如下:





变量名	类型	静态	数组	备注
列宽	整数型			

如果真 (输入框 (“请输入新列宽 (单位0.1毫米)”, “新列宽”,  
列宽, #输入整数) 且 列宽 > 0)  
数据源1.置列宽 (表格1.取光标列号 (), 表格1.取选择列数 (), 列宽)  
设置修改 ()

#### (10) “添加( )”方法

调用格式: <逻辑型> 对象. 添加 (数据源或数据提供者, [首行行号], [欲添加行数])  
将指定其他数据源或者数据提供者中的数据添加到本数据源的尾部。成功返回真, 失败返回假。

在“添加数据”菜单中的相关程序代码如下:

如果 (数据源2.从文件读 (通用对话框1.文件名) = 假)  
数据源2仅在此处使用  
信息框 (“打开欲添加文件失败!”, #错误图标, “错误”)  
数据源1.添加 (数据源2, , )  
设置修改 ()  
数据源2.初始尺寸 (1, 1)  
因为使用的是通用提供者2, 所以可以初始化尺寸. 在此处清除通用提供者2内的数据.

由于数据源 1 保存有数据, 所以不能通过“数据源 1.从文件读( )”读入新的数据。只有把新的数据库文件读入到另一个数据源中, 然后添加到数据源 1 中。

#### (11) “置类型( )”方法

调用格式: <无返回值> 对象. 置类型 (行号, 列号, [行数], [列数], [类型])  
设置数据源中指定单元格的数据类型。注意如果数据源所使用的数据提供者不支持此特性, 本命令将被忽略。

在“文本型”菜单中的相关程序代码如下:

数据源 1. 置类型 (表格 1. 取光标行号 (), 表格 1. 取光标列号 (), 表格 1. 取选择行数 (), 表格 1. 取选择列数 (), #文本型)

其他“数据源”组件有设置作用的方法使用格式与“置类型( )”方法一样, 具体使用和方法解释请参考“易之表.e”和易语言帮助。

#### (12) “删线条( )”方法

调用格式: <无返回值> 对象. 删线条 (行号, 列号, [行数], [列数], [线条类型])  
将数据源中指定范围内单元格内的线条去除。注意如果数据源所使用的数据提供者不支持此特性, 本命令将被忽略。

注意: 当“删线条( )”方法的“线条类型”与“加线条( )”方法的“线条类型”不一致时, 线条无法被删除。



以上是有关数据源的常用方法。如果遇到一些不熟悉的方法，可按下 F1 键，以查看即时的帮助。还有一些其他的功能，将在表格组件中进行介绍。

### 11.6.3 表格

#### 1. 表格的属性

##### (1) “数据源”属性

指定与表格相关联的数据源组件。在属性面板中设置本属性时，如果窗体上已经放置了某个（或多个）数据源组件，则会以下拉列表的形式列出，只需选择其中之一即可。如果要在程序中用代码的方式为本属性赋值，只需赋数据源组件名称的文本形式即可。实例代码如下：

表格 1. 数据源 = “数据源 1”

数据源属性是表格组件最重要的属性，不设置该属性，表格基本上没有存在的必要。

##### (2) “缩放比”属性

整数型，指定表格在显示数据时所采用的显示比例，可以是 20 到 1000 内的任意整数值，默认值是 100。

实例代码如下：

表格 1. 缩放比 = 50

如果要以页面的等宽显示，可以直接使用代码如下：

表格 1. 等宽缩放 ( )

##### (3) “表格线颜色”、“背景颜色”属性

打开书中例程“表格属性.e”。如图 11-12 所示。

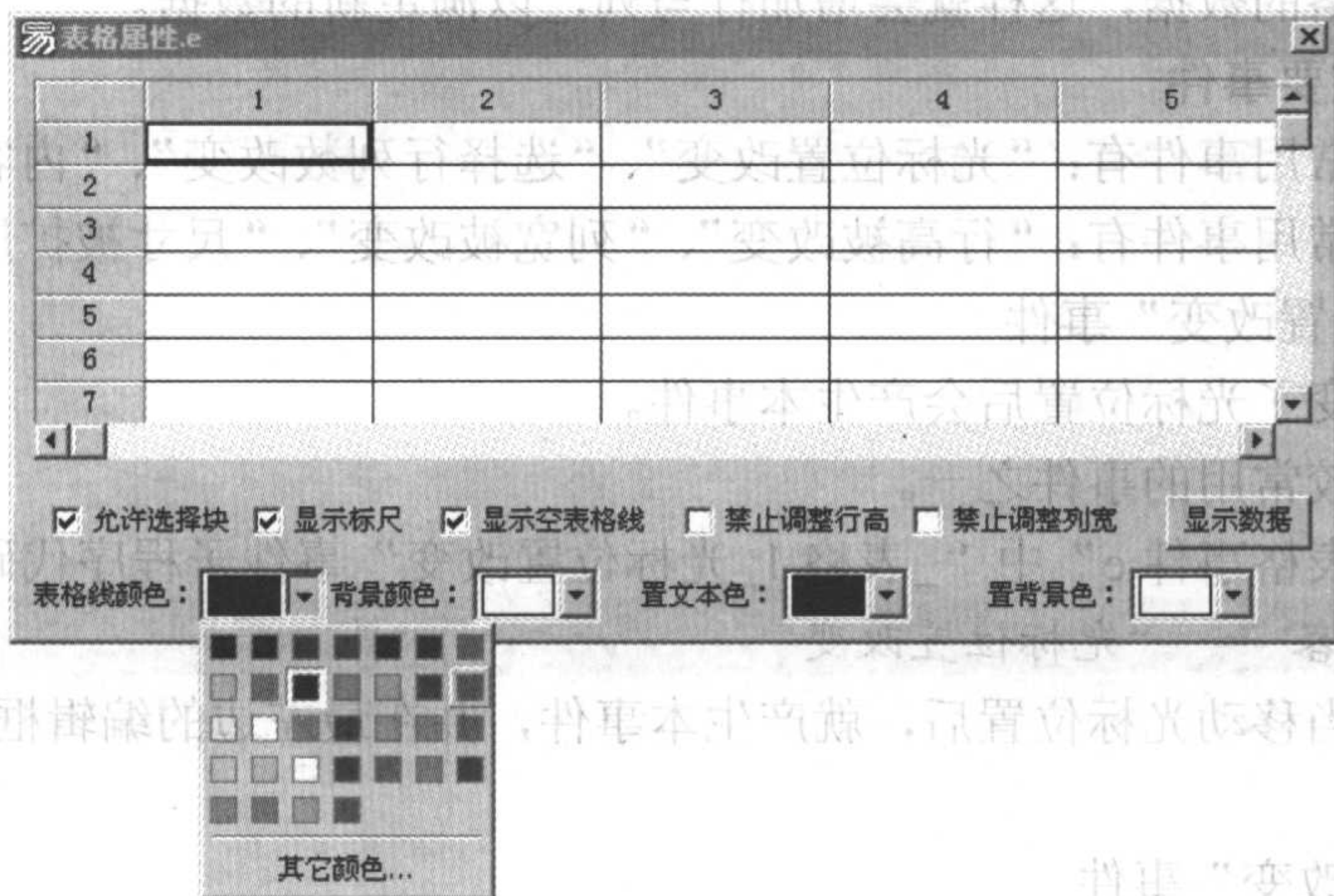


图 11-12 改表格线颜色

“\_颜色选择器 1\_颜色被改变”事件子程序代码如下：

数据源 1. 加线条 (1, 1, 数据源 1. 取行数 ( ), 数据源 1. 取列数 ( ), #左边框 + #上边框 + #右边框 + #下边框 + #水平线 + #垂直线)

表格 1. 表格线颜色 = 颜色选择器 1. 颜色





上面代码中可以看到，一定要先为表格线中加上线条，才能再改表格线颜色。而加线条是由数据源加入的。

“\_颜色选择器 2\_颜色被改变”事件子程序代码如下：

表格 1. 背景颜色 = 颜色选择器 2. 颜色

大家要问，如果要改变其中的文字应该怎么办呢？请看“\_颜色选择器 3\_颜色被改变”事件子程序，程序代码如下：

数据源 1. 置文本色 (表格 1. 取光标行号 (), 表格 1. 取光标列号 (), 表格 1. 取选择行数 (), 表格 1. 取选择列数 (), 颜色选择器 3. 颜色)

在这里，使用的是数据源组件，而不是表格组件，因此一定要注意。

单独改变其中某一个单元格的背景色，请看“\_颜色选择器 4\_颜色被改变”事件子程序，程序代码如下：

数据源 1. 置背景色 (表格 1. 取光标行号 (), 表格 1. 取光标列号 (), 表格 1. 取选择行数 (), 表格 1. 取选择列数 (), 颜色选择器 4. 颜色)

为表格加表格线，程序代码如下：

数据源 1. 加线条 (1, 1, 数据源 1. 取行数 (), 数据源 1. 取列数 (), #左边框 + #上边框 + #右边框 + #下边框 + #水平线 + #垂直线)

## (3) “允许粘贴扩展”属性

指定当粘贴表格数据时，如果现行表格尺寸无法容纳，是否允许自动扩展。

例如一个 2X2 单元格的表格插入一个 4X4 单元格的表格，即会自动增加不足的单元格。

如例程“易之表.e”中“\_添加数据\_被选择”事件中的程序代码，只有此属性为真时才有效。程序代码将一个新的易之表文件读入到数据源 2，并添加到数据源 1 中，由于会增加大于现有表格的数据，这样就会增加行与列，以满足新的数据。

## 2. 表格的重要事件

表格组件的常用事件有：“光标位置改变”、“选择行列数改变”、“内容被改变”。

表格组件不常用事件有：“行高被改变”、“列宽被改变”、“尺寸被扩展”。

### (1) “光标位置改变”事件

当操作者改变了光标位置后会产生本事件。

本事件是比较常用的事件之一。

书中例程“表格事件.e”中“\_表格 1\_光标位置改变”事件子程序代码如下：

编辑框 1. 内容 = “光标位置改变”

运行例程，当移动光标位置后，就产生本事件，并在最左边的编辑框 1 中显示提示信息。

### (2) “内容被改变”事件

当操作者修改了单元格内容时即产生此事件。

书中例程“表格事件.e”中“\_表格 1\_内容被改变”事件子程序代码如下：

编辑框 1. 内容 = “内容被改变”

运行例程，当在一个单元格中输入内容后，就产生本事件，并在最左边的编辑框 1 中显示提示信息。

可以看到，这个事件子程序会自动产生名称为“行号”、“列号”、“行数”、“列数”的



参数，以让编程者利用这些参数获取内容被改变的单元格位置。

其他事件的解释请参考易语言帮助和例程“表格事件.e”。

### 3. 表格的专有方法

#### (1) “取光标行号( )”、“取光标列号( )”方法

功能：取表格中“光标所在的单元格”所处的行号或列号。如果调用这两个方法时，已经有多个单元格被同时选择，则返回所有被选择的单元格中最左上角的那个单元格的行号或列号。

实例代码如下：

行坐标 = 表格 1. 取光标行号( )

列坐标 = 表格 1. 取光标列号( )

这两个命令在前面已多次用到，这里不多说了。

#### (2) “取选择行数( )”、“取选择列数( )”方法

功能：取表格中被选择的单元格的行数或列数。

书中例程“表格属性.e”。

其中使用“\_表格 1\_光标位置改变”事件子程序来显示光标行列号。

编辑框 1. 内容 = “光标位置改变”

编辑框 2. 内容 = 到文本 (表格 1. 取光标行号 ( ))

编辑框 3. 内容 = 到文本 (表格 1. 取光标列号 ( ))

使用“\_表格 1\_选择行列数改变”事件在编辑框中显示选择的行列数。程序代码如下：

编辑框 1. 内容 = “选择行列数改变”

编辑框 4. 内容 = 到文本 (表格 1. 取选择行数 ( ))

编辑框 5. 内容 = 到文本 (表格 1. 取选择列数 ( ))

#### (3) “选择( )”、“全部选择( )”、“复制( )”、“全部复制( )”、“粘贴( )”、“粘贴到光标处( )”方法

对单元格的选择、复制、粘贴操作。需要指定欲操作的一个或多个相邻的单元格。

一般都是由鼠标完成选择、复制、与粘贴的功能，提供命令只是为了使用右键弹出菜单来完成工作。

书中例程“表格命令.e”使用弹出菜单操作这些命令。

使用“\_表格 1\_鼠标右键被按下”事件子程序弹出菜单。相关程序代码如下：

弹出菜单 (小菜单, , )

菜单项“\_选择\_被选择”事件子程序代码如下：

表格 1. 选择 (表格 1. 取光标行号 ( ), 表格 1. 取光标列号 ( ), 表格 1. 取选择行数 ( ), 表格 1. 取选择列数 ( ))

其他菜单项事件子程序分别用到对应的表格方法。具体应用请参考例程。

#### (4) “打印( )”、“打印预览( )”方法

功能：打印表格，打印表格前的预览。

打印预览的功能可以在打印前预览最终打印效果，以防打印出错。

应用实例代码如下：





表格 1. 打印 ( )

表格 1. 打印预览 ( )

#### 4. 表格组件使用中出现的问題

(1) 怎样在程序中动态改变表格的列宽使它正好显示字段内容

需要在与“表格”相关联的“数据源”组件中设置。例如，表格 1 的数据源为“数据源 1”，则可以通过下列语句来实现。

数据源 1. 置列宽 (3, 1, 500) //将表格 1 的第三列设置宽度为 50 毫米。

(2) 如何将查询到的多条记录显示在表格中？实例代码如下：

变量名	类型	静态	数组	备注
计次变量1	整数型			
计次变量2	整数型			
行数	整数型			
列数	整数型			

数据源1. 清除 (1, 1, , )

--- 计次循环首 (行数, 计次变量1)

--- 如果真 (查找 (读 () = " " ))

数据源1. 添加行 ()

--- 计次循环首 (列数, 计次变量2)

数据源1. 置文本 (数据源1. 取行数 (), 列数, )

--- 计次循环尾 ()

跳过 ()

--- 计次循环尾 ()

(3) 如何进行打印设置？

打印设置可使用如下命令完成：

数据源 1. 打印设置 ( )

代码被执行时就会弹出打印设置对话框。可以调整页边距，纸张类型等。

其他问题请参看“易之表.e”以及“易之表增强版.e”程序。

#### 11.6.4 数据库相关组件的应用

1. 前面有很多组件都提供“数据源”属性，大家可以综合看一下这些组件是如何与数据源进行连接编程的，如编辑框、标签、图片框、组合框、列表框都支持数据源。

书中例程“列表框和组合框数据绑定演示.e”。如图 11-13 所示。



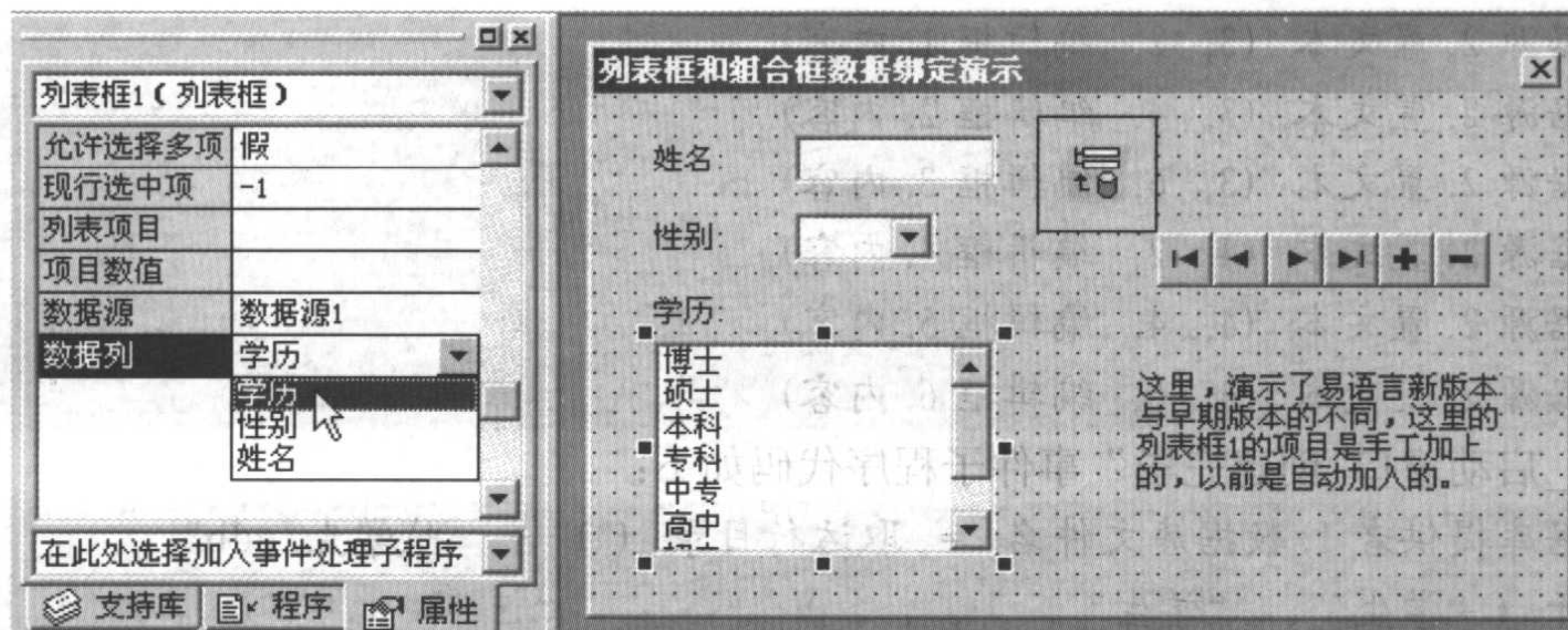


图 11-13 “列表框和组合框数据绑定演示.e”数据列的绑定

或者新建一个易程序，添加几个组件。数据库提供者 1，在“数据库文件名”属性中输入一个数据库的文件名。

数据源 1 的“数据提供者”属性选择“数据库提供者 1”。

编辑框 1 的“数据源”属性选择“数据源 1”，“数据列”属性选择“姓名”。

组合框 1 的“数据源”属性选择“数据源 1”，“数据列”属性选择“性别”。

列表框 1 的“数据源”属性选择“数据源 1”，“数据列”属性选择“学历”。

## 2. 应用到打印

(1) 如何在表格下角打印页码，解决方法如下：

假设表格是连接到“数据源 1”上，方法如下：

局部变量    打印设置信息    数据类型    打印设置信息

打印设置信息 = 数据源 1. 取打印设置 ( )

打印设置信息. 页号位置 = #下右页号

数据源 1. 置打印设置 (打印设置信息)

(2) 如何使用表格组件实现套打的功能？

表格组件不仅可以直接输入数据计算，也可以制作表格套打的模板，以后就可以按一定的模式进行套打。

书中例程“表格组件模板的应用.e”，这是一个打印学生情况表格的例程。

a. 启动窗口中“数据库提供者 1”组件的“数据库文件名”属性设置为数据库文件“学生.edb”，此数据库文件必须和当前易程序在同一目录下，否则设置为数据库文件的完全路径。

将“数据源 1”的“数据提供者”属性设置为“数据库提供者 1”。

将“编辑框”组件的“数据源”属性全部设置为“数据源 1”，分别将“数据列”属性选择为“学号”、“姓名”、“性别”、“年龄”、“身高”、“体重”。

为了更清楚一些，可以在编辑框前面加上标签文字说明。

将“数据源 2”的“数据提供者”属性设置为“通用提供者 1”。

将“表格”组件的“数据源”属性设置为“数据源 2”。

b. “\_数据源 1\_当前记录改变”事件子程序代码如下：





数据源 2. 置文本 (3, 2, 编辑框 1. 内容)  
 数据源 2. 置文本 (3, 4, 编辑框 2. 内容)  
 数据源 2. 置文本 (3, 6, 编辑框 3. 内容)  
 数据源 2. 置文本 (4, 2, 编辑框 4. 内容)  
 数据源 2. 置文本 (4, 4, 编辑框 5. 内容)  
 数据源 2. 置文本 (4, 6, 编辑框 6. 内容)

“\_启动窗口\_创建完毕”事件子程序代码如下:

```
数据库提供者 1. 数据库文件名 = 取运行目录 ( ) + “\学生.edb”
打开 ( “学生”, “学生”, , , , )
数据源 2. 数据提供者 = “通用提供者 1”
如果真 (数据源 2. 从文件读 (取运行目录 ( ) + “\表格模板.grd”) = 假)
    *备注。这次从文件读入表格样板到基本情况提供者, 用作演示模板的不同来源
    信息框 ( “找不到学生基本情况表格模板!”, 0, “错误”)
    销毁 ( )
    返回 ( )
如果真结束
学生表格. 表格线颜色 = #黑色
_数据源 1. 当前记录改变 ( )
```

在这里, “学生.edb”是一个含有很多记录的数据库, 而“表格模板.grd”是一个表格模板, 程序中将数据库中的内容读入到模板中显示出来。如果想打印, 可以再加入一个按钮, 改按钮标题为“打印表格”。此按钮的被单击事件程序代码如下:

```
如果真 (数据源 2. 打印设置 ( ))
    表格 1. 打印 ( )
如果真结束
```

### 11.6.5 制作表格模板

报表文件是一种表格形式的文件, 在数据的显示、打印和保存等方面发挥重要作用。在易语言中使用的报表文件是后缀为 grd 格式的表格模板文件。

跟易语言的表格组件相似, 表格模板文件主要是作为数据处理者来使用, 但又可以自由设置表格的大小、线条、位置、字体等, 可以制作跟现实生活中的表格一样的各种形式的表格, 这一点易语言的表格组件是无法办到的, 在数据库应用中经常用到表格文件。表格模板本身可以作为一种数据, 通过数据源读入到表格组件中并显示。

1. 易语言的表格模板文件制作工具就是易语言系统中附带的报表编辑器 (易之表)。报表的编辑方法跟我们用网页制作工具编辑网页表格的方法相似, 下面简单介绍一些编辑报表文件的方法。

(1) 打开报表编辑器。

打开易语言, 选择菜单“编辑”→“报表编辑器”。

(2) 打开或新建报表文件。



选“打开”可以打开已有的报表文件，选“新建”可以建立一个新报表文件。

(3) 删除行或删除列。

选中要删除的行，再选菜单“编辑”→“行列”→“删除行”。删除列的方法与此类似。同样我们也可以添加行或列。

(4) 合并或分解单元格。

将几个单元格合并成一个大的单元格，先选中需要合并的单元格，然后选鼠标右键菜单“组合”。选鼠标右键菜单“分解”可以分解单元格。

(5) 将表格虚线设为实线。

选中要设置实线的表格（可以选中全部表格），选菜单“单元格”→“边框或线条”→“添加全部表格线”。

(6) 对齐文字。

要对齐表格内的文字，选中表格，再选菜单“单元格”→“对齐方式”，再选择哪种对齐方式。

(7) 设置字体。

选菜单“单元格”→“字体”，设置各单元格内容的字体。

(8) 调整表格大小。

用鼠标拉动表头线或左端表线，可以调整表格大小。

(9) 设置单元格为图片类型。

选中单元格，选菜单“单元格”→“单元格类型”→“图片数据型”，设置此单元格位置放置的是图片。

(10) 保存文件。

选菜单“文件”→“保存”，将记录为空的表格模板保存为 grd 格式的文件，如“学生成绩报表.grd”。

2. 用报表编辑器打开“工资报表.grd”。由于表头行数被设置为 2 行，所以无法用光标选中单元格进行编辑。必须通过编辑菜单上的置表头行数重新设置。当模板中的单元格可被编辑时，用易之表提供的各项功能就可以修改模板结构。

表格模板有两种使用方法。

(1) 在“工资管理.e”例程中，使用“数据源.从文件读( )”命令先把模板文件“工资报表.grd”添加到数据源的前端，然后用“数据源.添加行( )”命令添加记录，同时把数据库的数据置到新加的空记录上。这样数据源中就将模板文件和数据库文件合并成一个文件，通过“表格.打印预览( )”命令预览合并后所有数据库的记录。

(2) 在“学生管理系统.e”例程中，“base.grd”模板文件被用来显示数据库的记录，一次只能显示一条数据库记录。每个模板表格单元显示数据库的不同字段的数据，相当于在窗体上添加多个拥有“数据源”属性的组件。





## 11.7 本章小结

易数据库的功能与规模大体相当于 FOXBASE, 可通过易语言指令进行操纵、直接访问、存储、查询本数据库, 运行效率高, 适用于小型企事业单位及家庭等应用。

该数据库目前已支持绝大部分常用操作, 譬如索引、查找、过滤、排序、替换、加密、多用户访问共享冲突规避等等。并提供了相应的可视化建库和记录录入接口软件。

大家也可以进行以下练习:

1. 自行编写一个使用易数据库的小型通讯录软件。含增加、删除、修改、查找等功能。
2. 在表格中显示易数据库记录内容, 并用表格模板打印出来。



## 第十二章 外部数据库调用

易语言可以操作支持 ODBC 和 ADO 的外部数据，比微软的 Access 数据库（文件后缀名为 mdb）、dBase 数据库（文件后缀名为 dbf）、Visual FoxPro 数据库（文件后缀名为 dbc）、MS SQL Server 大型数据库、MYSQL 大型数据库等。

本章将对易语言外部数据库的调用与操作进行详细的介绍。

### 12.1 外部数据库相关知识

#### 12.1.1 易外部数据库组件简介

易语言提供的操作外部数据库组件自 3.6 版推出以后，已经增加到了 4 种，包括“外部数据库”、“外部数据库提供者”、“数据库连接”和“记录集”。

“外部数据库”组件和“外部数据提供者”组件，可以直接将外部数据库绑定到 ODBC 数据源。并对外部数据库直接进行操作。

“数据库连接”和“记录集”组件，用作使用 ADO 直接对外部数据库进行操作。

#### 12.1.2 ODBC 与 ADO 简介

##### 1. ODBC 简介

ODBC（Open Database Connectivity）即开放式数据库互连标准。ODBC 是一种客户端连接后台数据库的技术。

ODBC 提供了一套标准函数，可以较容易的访问本地或远程数据库。

一般 ODBC 由以下三部分组成：

- ODBC 驱动管理程序
- 一个或多个 ODBC 驱动程序
- 一个或多个 ODBC 数据源

ODBC 的结构如图 12-1 所示。



图 12-1 ODBC 结构图





如图 12-1 所示，客户应用程序使用 ODBC 数据源，ODBC 数据源通过 ODBC 驱动程序管理与特定的 ODBC 驱动程序联系起来，然后通过此 ODBC 驱动程序来访问本地或远程数据库。

### ODBC 数据源的概念

ODBC 数据源又叫 DSN，他把客户应用程序所要使用的驱动程序、数据库、用户名和口令等信息组合起来，供客户端程序使用。

ODBC 数据源 (DSN) 有三种类型：

- 用户 DSN：只能由配置该 DSN 的用户使用或只能在当前的计算机上使用。
- 系统 DSN：在多用户系统中，可被任何用户使用。另外，如果用户要建立 Web 数据库应用程序，应使用此数据源。
- 文件 DSN：与系统 DSN 相似，但用户可以在其他计算机上对其进行使用。

易语言在使用 ODBC 组件时，就需要用到文件 DSN。在使用数据源时，首先要对数据源进行配置，后面在介绍 ODBC 例程时将会介绍如何配置 ODBC 数据源，以及如何创建一个文件 DSN。

配置 ODBC 数据源，可以使用“控制面板”→“管理工具”→“数据源 (ODBC)”工具来配置数据源 (Windows XP 系统)。

## 2. ADO 简介

ADO (ActiveX Data Object) 是微软公司为最强大的数据访问范例 OLE DB 而设计的，是一个便于使用的应用程序层接口。可以为任何数据源提供高性能的访问。ADO 在 Internet 方案中使用最少的网络流量，并且在前端和数据源之间使用最少的层数。

ADO 对象是以分层结构组织的，如图 12-2 和 12-3 所示。

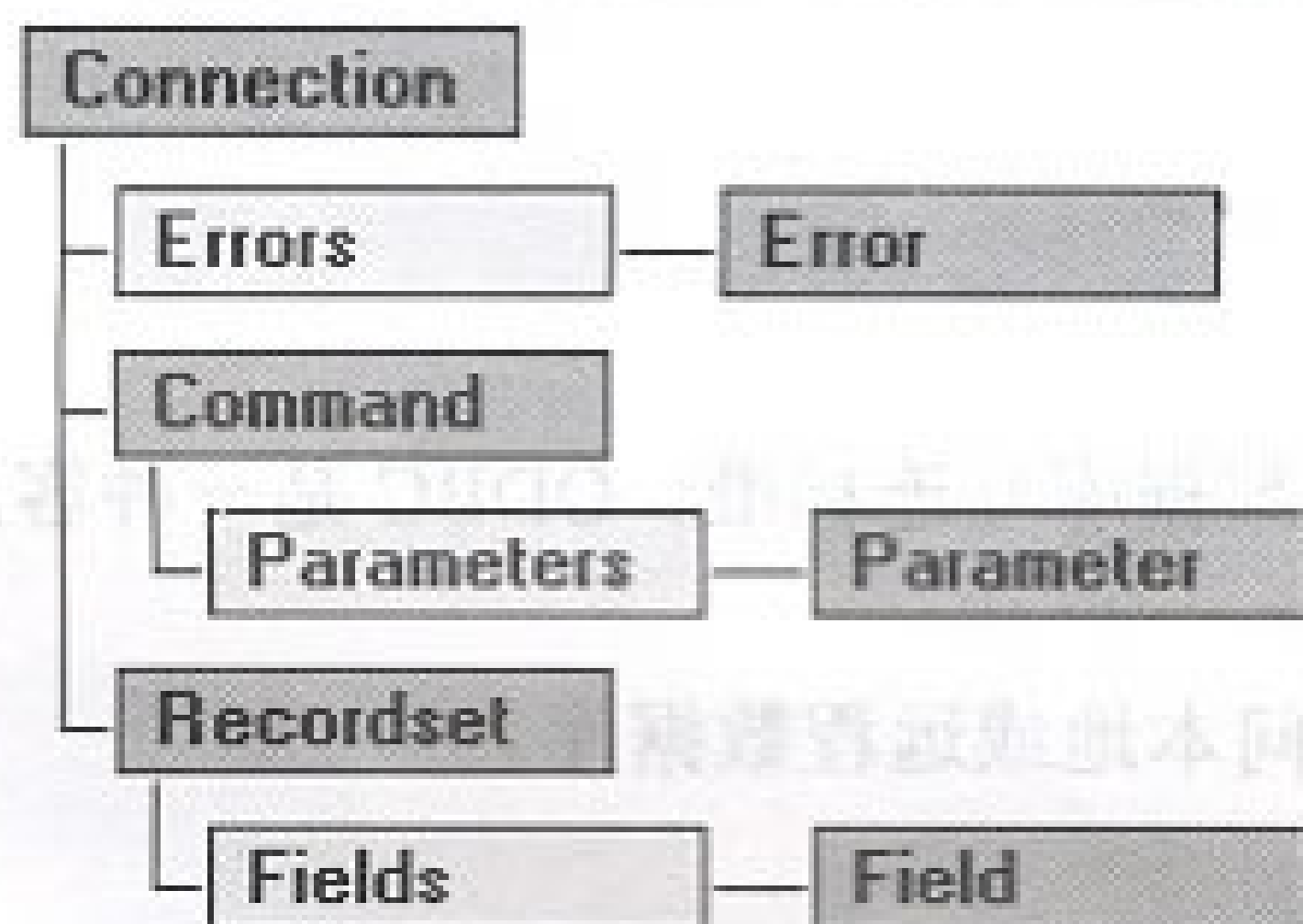


图 12-2 ADO 分层结构图 1

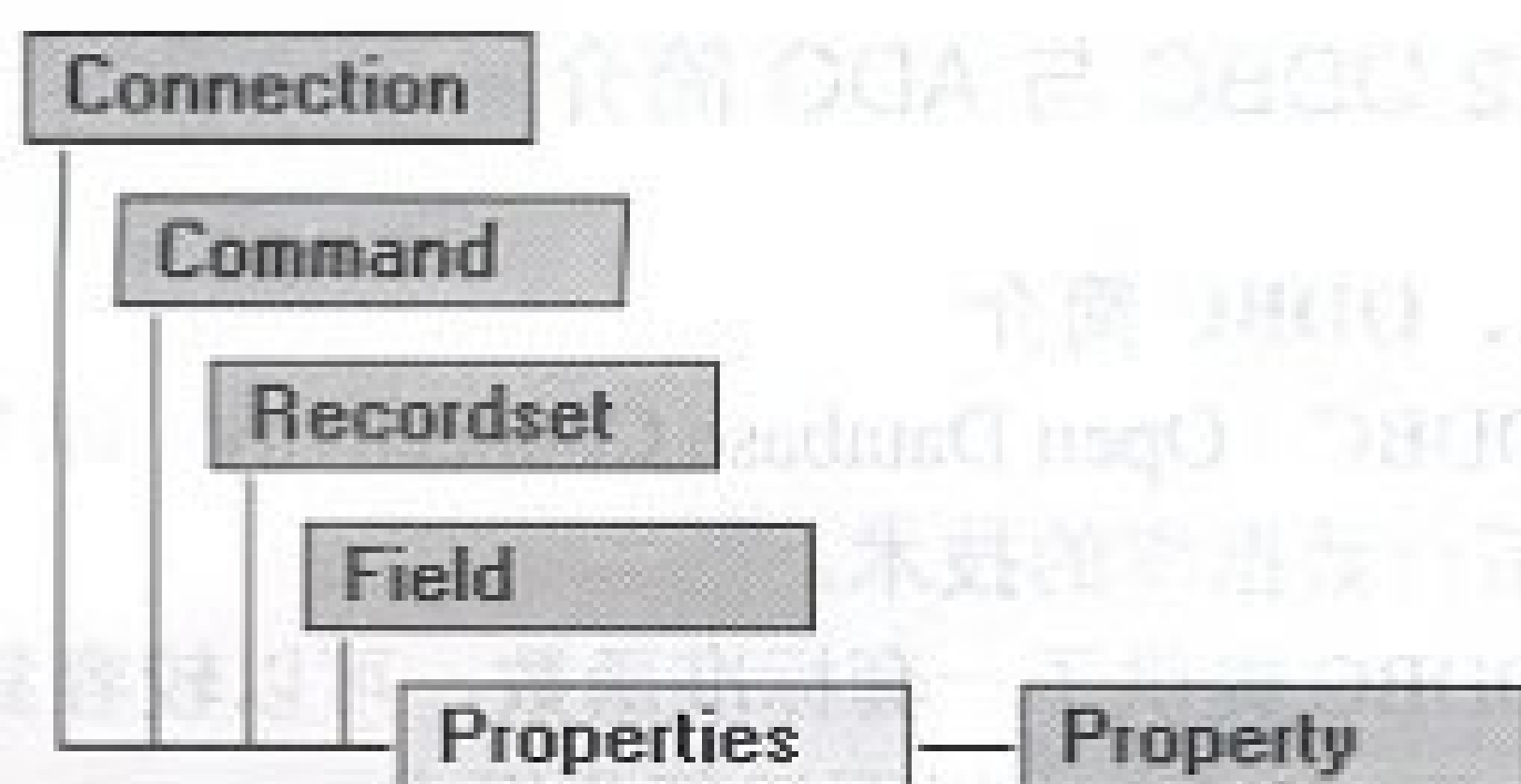


图 12-3 ADO 分层结构图 2

上面两个结构图中包括了 ADO 对象模型中的七个主要对象，分别是 Connection 对象、Command 对象、Recordset 对象、Fields 对象、Parameters 对象、Errors 对象和 Properties 对象。这些对象拥有可以操作数据的“方法”，以及表示数据某些特性或控制某些对象方法行为的“属性”。

易语言中使用“数据库连接”和“记录集”组件以 ADO 方式操作外部数据库。

以 ADO 方式操作数据库比 ODBC 方式操作数据库有更高的效率。



### 12.1.3 SQL 语言简介

SQL 语言（结构化查询语言）是对关系数据库进行操作的标准语言，例如：Access、SQL Server、Oracle 等大型数据库都支持 SQL 语言。所以学习操作外部数据库前，建议先对 SQL 语言作一些了解。

SQL 语言具有功能丰富、使用方便灵活、语言简洁等优点。在 1986 年 10 月，美国国家标准局（ANSI）批准了 SQL 作为关系数据库语言的美国国家标准，此后不久，国际化标准组织（ISO）也做出了同样的决定，使 SQL 成为了国际标准。

SQL 语言的功能包括查询、操纵、定义和控制四个方面。SQL 语言的每个功能都由若干条指令组成，每条指令表示对数据库的一种操作。

### 12.1.4 常用的 SQL 语句

常用的 SQL 语句包括：

数据库查询：SELECT

数据定义：CREATE，DROP

数据操纵：INSERT，UPDATE，DELETE

数据控制：GRANT，REVOKE

#### 1. 查询语句 SELECT

SELECT 语句是很常用的 SQL 语句，语法也很简单，但是通过变化查询条件和查询方式，SELECT 语句可以完成多种查询任务，格式如下：

SELECT 目标字段

FROM 基本表

[WHERE 条件表达式]

[GROUP BY 字段名 1[HAVING 内部函数表达式]]

[ORDER BY 字段名 2 ASC/DESC]

语句含义：根据 WHERE 子句中的条件表达式，从基本表中找到满足条件的记录，按 SELECT 子句中的目标字段，选出记录中的分量形成结果。如果有 ORDER 子句，则结果根据指定字段名 2 按升序（ASC）或降序（DESC）排列。GROUP 子句将结果按字段名 1 分组，只有满足 HAVING 子句中内部函数表达式的组才会输出到结果表中。

在 WHERE 子句中可以有以下的条件选择：参见表 12-1 WHERE 子句中常用运算符及意义。


表 12-1 WHERE 子句中常用运算符及意义

运算符	意义
=	等于
>	大于
<	小于
>=	大于等于
<=	小于等于





表 12-1 WHERE 字句中常用运算符及意义

	不等于
LIKE	类似于（支持通配符）
NOT LIKE	不类似于（支持通配符）
AND	AND 连接的多个条件必须同时满足
OR	OR 连接的多个条件至少满足其中 1 个
NOT	排除后跟条件，例如：小明 NOT 小强

进行简单的查询：

假设已经新建了一个学生成绩表，如表 12-2 所示。

表 12-2 学生成绩表

学号	姓名	语文	数学	英语	班级
1	明天	78	76	68	1年1班
2	张强	89	87	76	1年1班
3	王一	99	97	78	1年2班
4	李杰	67	78	78	1年2班
5	朱名	99	90	95	1年2班
6	赵直	88	90	91	1年3班
7	孙新	87	79	81	1年3班

例 1：查询表中 1 年 2 班的所有同学，则使用 SQL 语句：

```
SELECT * FROM 学生成绩表 WHERE 班级='1 年 2 班'
```

语句中的“\*”代表查询和返回所有字段，FROM 后面的“学生成绩表”是一个关键字，指定所查询的表的名称。WHERE 后面的表达式，是要查询的条件。

在易语言中，如果使用“外部数据库”组件，可以用下面代码执行上面的 SQL 语句：

外部数据库 1. 查询（“SELECT \* FROM 学生成绩表 WHERE 班级='1 年 2 班'”）

SQL 语句在易语言中的使用后面会详细介绍，这里只举一个简单的例子。

例 2：查询“学生成绩表”中语文成绩小于 90 并且大于 70 的所有人，并返回“姓名”字段和语文成绩字段的记录内容，最后按语文成绩的升序排列查找结果。使用 SQL 语句：

```
SELECT 姓名, 语文 FROM 学生成绩表 WHERE 语文>70 and 语文<90 ORDER BY 语文  
ASC
```

查询的结果将只返回符合条件记录的姓名和语文字段的内容，由于有 ORDER BY 子句：“ORDER BY 语文 ASC”，所以返回的记录将会按语文成绩的升序排列。

例 3：查询“学生成绩表”中学号在 3 号至 6 号的学生成绩。使用 SQL 语句：

```
SELECT * FROM 学生成绩表 WHERE 学号 BETWEEN 3 AND 6
```

例 4：查询“学生成绩表”中姓“李”的学生成绩，使用 SQL 语句如下：

```
SELECT * FROM 学生成绩表 WHERE 姓名 LIKE '李%'
```



例 4 中使用了 LIKE 语句, LIKE 语句的格式是:

字段名 LIKE 指定字符串

其中的字符串有以下几种形式:

“%”: 代表任意长度的任意字符。

“\_”: 下划线代表任意单个字符。

其他字符: 仅代表自身。

所以上面的 SQL 就查询了所有姓名中第一个字是“李”的学生记录。

## 2. 连接查询

如果查询中涉及了 2 个以上的表, 则称之为连接查询。在 WHERE 子句中引导的查询条件中字段的表示方法, 有些类似易语言中组件属性的表示方法, 表示为:

欲查询表. 字段名

例如, 当前有 2 个表, 第一个表是上面用到的表 12-2 学生成绩表, 另一个如表 12-3 学籍管理表。

表 12-3 学籍管理表

学号	姓名	家庭住址	联系电话	籍贯
1	明天	大连某区某街	82234124	山东
2	张强	沈阳某区某街	32432553	山东
3	王一	南京某区某街	23434555	湖南
4	李杰	吉林某区某街	34534223	湖北
5	朱名	大连某区某街	82345567	上海
6	赵直	北京某区某街	75346312	北京
7	孙新	天津某区某街	23576865	广东

例 1: 查询两个表中学号相同人的所有信息。使用 SQL 语句:

SELECT \* FROM 学生成绩表, 学籍管理表 WHERE 学生成绩表. 学号=学籍管理表. 学号

WHERE 子句中的条件, 两个字段进行比较, 那这两个字段必须具有可比性, 但不必相同, 也可以用=、>、<、>=、<=、<>连接。

例 2: 查询两个表中籍贯是“山东”的所有学生语文成绩。使用 SQL 语句:

SELECT 学生成绩表. 学号, 学籍管理表. 学号, 学生成绩表. 姓名, 学生成绩表. 语文, 学籍管理表. 籍贯 FROM 学生成绩表, 学籍管理表 WHERE 学生成绩表. 学号=学籍管理表. 学号 AND 学籍管理表. 籍贯=' 山东'

语句中通过使用 AND 逻辑词, 将两个条件连接, 可以查询到满足多个条件的记录。还可以使用 OR 逻辑词连接多个可选条件, 其中有一个条件符合即为满足条件。

## 3. 创建表

SQL 语言中的 CREATE TABLE 语句被用来建立新的数据库表格。CREATE TABLE 语句的使用格式如下:





## CREATE TABLE 表格名

(第 1 个字段名 第 1 个字段数据类型 [字段长度],  
第 2 个字段名 第 2 个字段数据类型 [字段长度],  
第 3 个字段名 第 3 个字段数据类型 [字段长度]  
...)

在 CREATE TABLE 后面加入所要建立的表格的名称, 然后在括号内顺次设定各列的名称, 数据类型, 以及可选的限制条等。

例如: 要使用 SQL 语句创建一个名为“学生成绩表”的数据库表格, 有 3 个字段, 第一个字段为“学号”, 整数型; 第二个字段为“姓名”, 文本型; 第三个字段为“英语”, 整数型。使用 SQL 语句:

```
CREATE TABLE 学生成绩表(学号 int, 姓名 text, 英语 int)
```

在易语言中, 如果已经用“外部数据库”组件打开了一个数据库, 要想在该数据库中创建上面的表, 可以使用代码:

```
外部数据库 1. 执行 (“CREATE TABLE 学生表(学号 int, 姓名 text, 英语 int)”, )
```

## 4. 修改表中记录

要修改表中已经存在的一条或多条记录, 应使用 UPDATE 语句。UPDATE 语句格式如下:

**UPDATE 被操作表名**

**SET 字段及内容[字段及内容,...]**

**[WHERE 条件表达式]**

在 UPDATE 后面是欲操作的表名, SET 后面是欲改变的字段及内容, 然后在 WHERE 子句中填写被改变记录要符合的条件。

例如, 前面的表 12-2 学生成绩表中, 将张强的英语成绩改为 99, 使用 SQL 语句如下:

```
UPDATE 学生成绩表 SET 英语=99 WHERE 姓名='张强'
```

当然, UPDATE 语句也可以修改多个表, 多个字段的内容, 比如: 要将表 12-2 学生成绩表和表 12-3 学籍管理表中, 姓名为张强的记录的英语成绩改为 99, 籍贯改为“湖南”, 使用以下 SQL 语句:

```
UPDATE 学生成绩表, 学籍管理表 SET 学生成绩表. 英语=99, 学籍管理表. 籍贯='湖南'  
WHERE 学生成绩表. 姓名='张强' and 学籍管理表. 姓名='张强'
```

## 5. 删除表中记录

要从表中删除一个或多个记录, 需要使用 DELETE 语句。给 DELETE 语句提供 WHERE 子句。WHERE 子句用来选择要删除的记录。格式如下:

**DELETE [FROM 被操作表名]**

**[WHERE 条件语句]**

FROM 语句后是欲删除记录的表名, WHERE 语句中填写要删除记录所符合的条件。

**注意:** 在没有指定 WHERE 子句时, DELETE 语句将删除表中的所有记录, 所以使用



的时候要十分注意。

例 1: 删除表 12-2 学生成绩表中, 姓名为“李杰”的记录, 使用 SQL 语句如下:

```
DELETE FROM 学生成绩表 WHERE 姓名='李杰'
```

例 2: 删除表 12-2 学生成绩表中, 姓名为“李杰”和“张强”的记录, SQL 语句如下:

```
DELETE FROM 学生成绩表 WHERE 姓名='李杰' OR 姓名='张强'
```

## 6. 插入记录

要在表中插入记录使用 INSERT 语句, 格式如下:

**INSERT [INTO] 被操作表名**

**(第 1 个添加记录的字段名,...最后 1 个添加记录的字段名)**

**[VALUES] (第 1 个添加的数据,...最后 1 个添加的数据);**

向表中插入数据, 要在关键字 **INSERT INTO** 之后紧跟着表名, 在一对括号中, 是以逗号分开的一系列的字段名; 然后在关键字 **VALUES** 之后跟着一系列用括号括起的数值。这些数值是要往表格中填入的数据, 它们必须与指定的字段名相匹配。

例如, 要在 12-2 学生成绩表中插入一个记录, 学号为 8, 姓名为“郑新新”, 英语成绩为 88, 数学成绩为 98, 语文成绩为 85。

```
INSERT INTO 学生成绩表 (学号, 姓名, 英语, 数学, 语文) VALUES (8, '郑新新', 88, 98, 85)
```

**注意:** INSERT 语句与 DELETE 语句和 UPDATE 语句有一点不同, 它一次只操作一个记录。然而, 有一个方法可以使 INSERT 语句一次添加多个记录。要做到这一点, 需要把 INSERT 语句与 SELECT 语句结合起来。

例如, 当前数据库有 2 个学生成绩表, 第一个表是表 12-2 学生成绩表, 另一个表 12-4 学生成绩表 2。将学生成绩表 2 中班级为 1 年 2 班的记录, 复制到表 12-2 学生成绩表中, 使用 SQL 语句:

```
INSERT INTO 学生成绩表 (姓名, 语文, 数学, 英语, 班级) SELECT 姓名, 语文, 数学, 英语, 班级 FROM 学生成绩表 2 WHERE 班级='1 年 2 班'
```

表 12-4 学生成绩表 2

学号	姓名	语文	数学	英语	班级
1	李新	78	89	98	1年1班
2	张路	76	77	87	1年2班
3	刘真	100	99	67	1年2班

## 7. 删除表

用 DROP TABLE 命令可以删除一个表格。其语法格式为:

**DROP TABLE 欲删除的表名**

例如, 将表 12-2 学生成绩表删除, 可以用 SQL 语句:

```
DROP TABLE 学生成绩表
```





## 8. 集合函数

前面学习了根据特定的条件从表中取出一条或多条记录。但是，假如想对一个表中的记录进行数据统计。例如，如果想统计存储在表中的一次民意测验的投票结果。或者想知道一个访问者在站点上平均花费了多少时间。要对表中的任何类型的数据进行统计，都需要使用集合函数。

Microsoft SQL 支持五种类型的集合函数。可以统计记录数目，平均值，最小值，最大值，或者求和。

### (1) 记录数目

函数 COUNT () 可以用来统计一个表中有多少条记录。

例如，要统计表 12-2 学生成绩表中，数学成绩 90 分以上的有几个学生，可以使用以下 SQL 语句：

```
SELECT COUNT(数学) FROM 学生成绩表 WHERE 数学>=90
```

### (2) 取平均值

使用函数 AVG ()，你可以返回一个字段中所有值的平均值。

例如，要统计表 12-2 学生成绩表中，英语成绩的平均值，可以使用 SQL 语句：

```
SELECT AVG(英语) FROM 学生成绩表
```

### (3) 求和

使用函数 SUM ()，你可以返回一个字段中所有值之和。

例如，有表 12-5 教师工资表，要计算出该月一共要支付多少工资，可以使用 SQL 语句：

```
SELECT SUM(工资) FROM 教师工资表
```

表 12-5 教师工资表

教师工号	教师名	工资
1	王校长	2000
2	李主任	1500
3	赵老师	1000
4	张老师	1000

### (4) 取最大值、取最小值

MAX () 函数可以返回一个字段的最大值，MIN () 函数可以返回一个字段的值。

例如，要取出表 12-2 学生成绩表中数学成绩最好的分数，使用 SQL 语句：

```
SELECT MAX(数学) FROM 学生成绩表
```

常用的 SQL 语句就介绍这些，如果还要使用更高级的 SQL 语句，可以查看相关书籍和资料，本书将不作深入介绍。



## 12.2 ODBC 连接数据库组件

易语言中 ODBC 连接数据库的组件包括：“外部数据库”和“外部数据库提供者”，下面将主要以 Access 数据库为例介绍 ODBC 连接外部数据库组件。

### 12.2.1 “外部数据库” 组件属性

“外部数据库”组件只有简单的“名称”、“备注”、“左边”、“顶边”、“宽度”、“高度”、“标记”属性。它主要通过外部数据库组件的方法对外部数据库进行操作。

### 12.2.2 “外部数据库” 组件重要方法

可在支持库面板中查找外部数据库组件的方法。如图 12-4 所示。

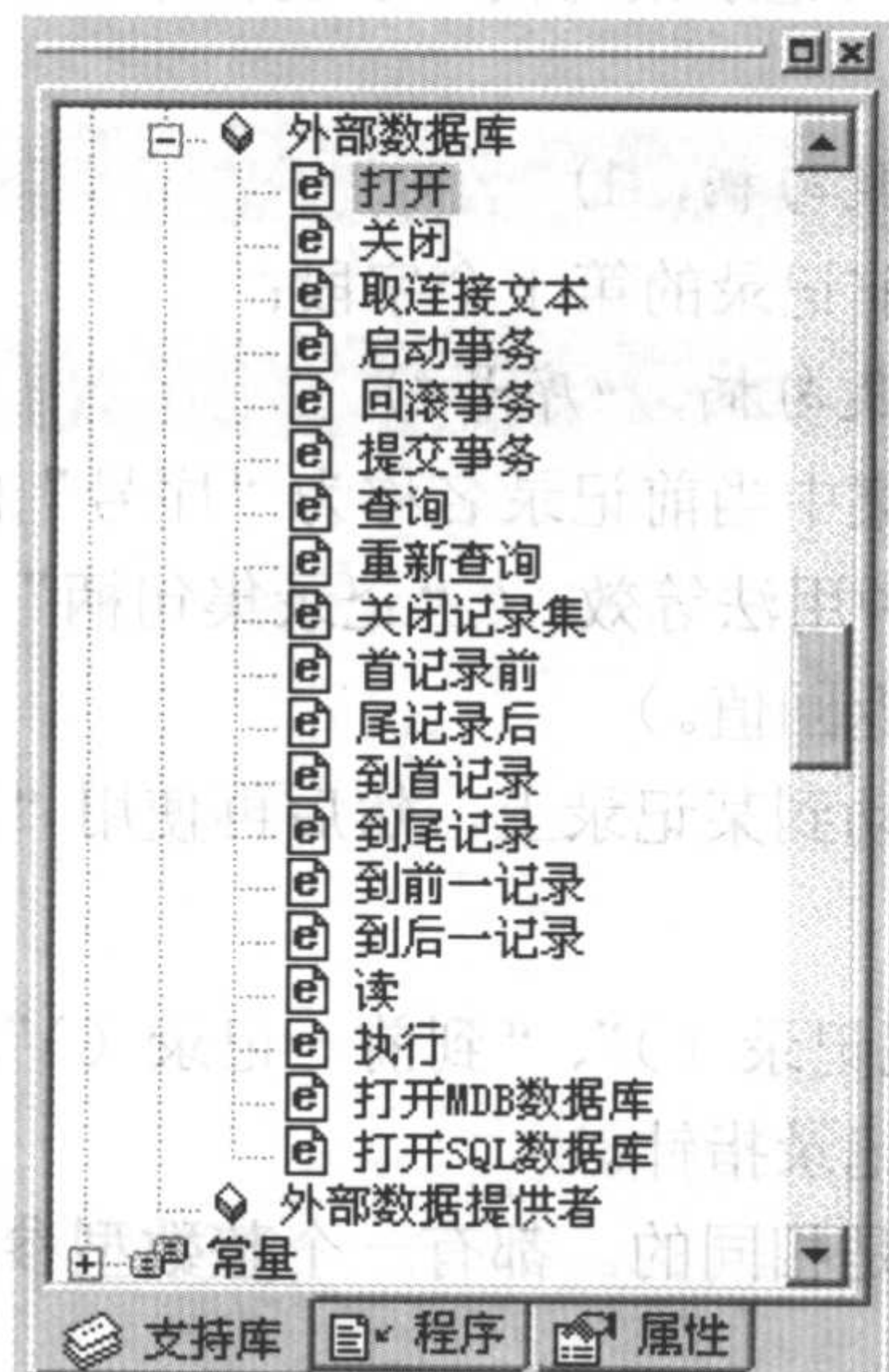


图 12-4 外部数据库组件的方法

#### 1. “打开 ()”、“关闭 ()” 方法

功能：打开或关闭指定的 ODBC 数据源。

应用实例：

外部数据库 1. 打开 ( , ) ' 打开，在使用外部数据库之前调用

外部数据库 1. 关闭 ( ) ' 关闭，不再使用外部数据库时调用

**注意：**“打开 ()”方法有两个参数，其中第一个参数是“ODBC 数据源连接文本”。关于该参数的使用将在后面介绍。

#### 2. “查询 ()”方法、“执行 ()”方法





功能：分别用于执行查询类和非查询类 SQL 语句。

应用实例：

(1) 例如执行查询类 SQL 语句：

```
记录集句柄 = 外部数据库 1. 查询("select * from books")
```

对当前被打开数据库进行数据查询(检索 books 表中的所有记录的所有字段)，将查询结果的记录集的句柄，赋值给**整数型**变量“记录集句柄”。

**注意：**以后不用本记录集时，应将本记录集关闭，方法是调用本组件的“关闭记录集 ()”方法，如：外部数据库 1. 关闭记录集(记录集句柄)。

(2) 例如执行非查询类 SQL 语句：

```
外部数据库 1. 执行("insert into books values (1,2,3)") //插入记录
```

### 3. “读 ()”方法

功能：读取指定记录集的当前记录处指定字段的数据内容。返回数据的数据类型与原外部数据库中的数据类型相对应。

语法：外部数据库名称.读 (记录集句柄, 字段名称或位置)

应用实例：

```
X = 外部数据库 1. 读(记录集句柄, 1)
```

上述读取指定记录集中当前记录的第 1 个字段；

```
X = 外部数据库 1. 读(记录集句柄, "序号")
```

上述语句为读取指定记录集中当前记录名称为“序号”的字段。如果数据库的第 1 个字段正好是“序号”，则这两种用法等效。(“记录集句柄”是整数型变量，是调用外部数据库组件的“查询”方法时的返回值。)

通常先将当前记录指针移动到某记录上，然后再使用“读 ()”的方法。移动记录指针的方法后述。

### 4. “到首记录 ()”、“到尾记录 ()”、“到前一记录 ()”、“到后一记录 ()”方法

功能：移动记录集的当前记录指针。

语法：这四个方法的语法是相同的。都有一个**整数型**参数，指定要操作的记录集；返回值表明本方法是否执行成功。

应用实例：

```
到首记录(记录集句柄)
```

```
到尾记录(记录集句柄)
```

```
到前一记录(记录集句柄)
```

```
到后一记录(记录集句柄)
```

“记录集句柄”是整数型变量，是调用外部数据库组件的“查询”方法时的返回值。

### 5. “首记录前 ()”、“尾记录后 ()”方法

功能：判断当前记录指针是否在第一个记录之前，或最后一个记录之后。

语法：这两个方法的语法是相同的。都有一个**整数型**参数，指定要操作的记录集；返回值表明当前记录指针是否在第一条记录之前，或在最后一条记录之后。



实例 1:

子程序: -查看前一记录按钮\_被单击

外部数据库 1. 到前一记录 (记录集句柄)

如果 (外部数据库 1. 首记录前 (记录集句柄))

信息框 ("前面已无记录!", 0, )

外部数据库 1. 到首记录 (记录集句柄)

否则

读取并显示当前记录 () ' 这是自定义函数

如果结束

实例 2:

子程序: -查看后一记录按钮\_被单击

外部数据库 1. 到后一记录 (记录集句柄)

如果 (外部数据库 1. 尾记录后 (记录集句柄))

信息框 ("后面已无记录!", 0, )

外部数据库 1. 到尾记录 (记录集句柄)

否则

读取并显示当前记录 () ' 这是自定义函数

如果结束

### 12.2.3 “外部数据库提供者” 组件

外部数据提供者组件除基本属性外, 还有两个重要的属性: “连接文本”、“查询 SQL”。外部数据提供者组件主要通过数据源组件对外部数据提供者中的数据进行操作。

#### 1. “连接文本”属性

“连接文本”属性为文本型, 用于设置外部数据库的 ODBC 连接文本。当在属性夹单击“...”按钮将弹出 ODBC 数据源配置对话框 (具体操作请参考后面的关于“ACCESS 数据库使用详解”内容)。

#### 2. “查询 SQL”属性

“查询 SQL”属性为文本型, 用作指定数据库中的数据表名或者用作查询记录集的 SELECT 类 SQL 语句。

## 12.3 ADO 操作数据库组件

易语言以 ADO 方式连接数据库的组件包括: “数据库连接”和“记录集”, 下面介绍这些组件重要的属性和方法。





### 12.3.1 “数据库连接”组件

#### 1. “数据库连接”组件属性

“数据库连接”组件重要的属性有：“最后错误”、“是否已连接”、“对象提供者”和“引擎版本”。这些属性都是只读属性，即只能读取它的内容而不能更改。

“最后错误”属性，可以获得最后出现的错误文本，是文本型属性。

“是否已连接”属性，为逻辑型。表示是否已经连接到数据库。

“对象提供者”属性，文本型。对象提供者名称。

“引擎版本”属性，为数据引擎版本，为文本型。

在程序中可以根据需要，读取这些属性的值。例如判断组件是否已经连接到了一个数据库，可以使用代码：

```
信息框 (选择 (数据库连接 1. 是否已连接, “已经连接到数据库”, “尚未连接到数据库”), 0, )
```

#### 2. “数据库连接”组件方法

“数据库连接”组件的方法，可以在支持库面板的“数据库操作支持库”中找到，如图 12-5 所示。

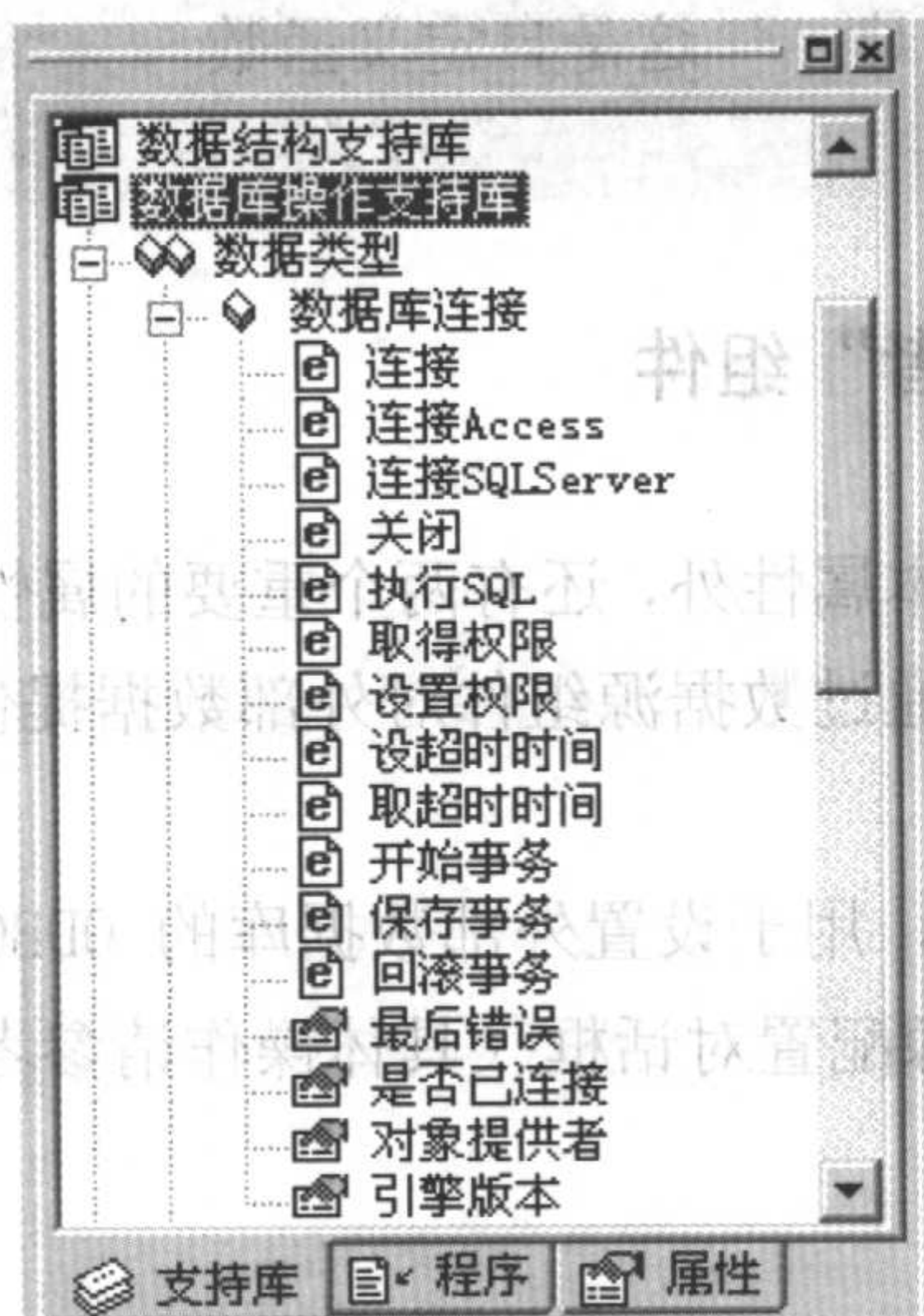


图 12-5 “数据库连接”组件方法

下面介绍“数据库连接”组件的一些常用方法：

##### (1) “连接 ()”方法

功能：用于连接到数据源，成功返回“真”，失败返回“假”。

在对外部数据库进行操作前首先要建立与数据库之间的连接。该方法的参数中填入连接文本。

连接一个 SQL Server 数据库代码如下：



数据库连接 1. 连接 (“DRIVER=SQL Server; SERVER=192.168.0.19; UID=user; PWD=123; WSID=BOOK9; DATABASE=test”)

连接文本中包括了连接数据库服务器、用户名、密码等信息。例如例句中的连接文本：“DRIVER”表示数据库服务器名、“SERVER”表示数据库的 IP 地址、“UID”表示用户名、“PWD”表示密码、“WSID”表示主机名、“DATABASE”表示操作的数据库名。

### (2) “连接 Access ()” 方法

功能：可以连接到指定目录中的 Access 数据库文件。

语法：数据库连接. 连接 Access (文件名, 密码)

第一个参数填写欲连接的数据库文件名，如果数据库有密码，第二个参数填写数据库密码。例如，连接到 C 盘下的“学生成绩”数据库：

连接 Access (C:\学生成绩.mdb, “”)

### (3) “连接 SQL Server ()” 方法

功能：可以连接到指定的 SQL Server 数据库。

语法：数据库连接. 连接 SQLServer (服务器名, 数据库名, 用户名, 密码)

应用实例：

数据库连接 1. 连接 SQLServer (“192.168.0.66”, “学生信息数据库”, “users”, “123456”)

代码中的用户名和密码是在 SQL Server 的企业管理器中创建的，后面将详细介绍。

### (4) “关闭 ()” 方法

功能：断开当前数据库连接。成功返回“真”，失败返回“假”。

数据库连接 1. 关闭 ()

### (5) “执行 SQL 方法 ()” 方法

功能：执行指定的查询、SQL 语句、存储过程等，执行成功返回真，否则返回假。

该方法的返回值是逻辑型，表示执行是否成功。例如，使用 SQL 语句为“学生信息数据库”创建一个“学生信息表”：

数据库连接 1. 执行 SQL (“CREATE TABLE 学生信息表(姓名 text, 年龄 int, 籍贯 text)”)

### (6) “取得权限 ()”、“设置权限 ()” 方法

功能：“取得权限 ()”方法取得访问及相关权限，并返回一个整数代表用户连接当前的权限。返回整数代表的意义，可以查看易语言帮助。

“设置权限 ()”方法设置访问及相关权限。

应用实例：

信息框 (数据库连接 1. 取得权限 (0, 0,))

此行代码可以使用信息框查看当前对数据库的操作权限。

数据库连接 1. 设置权限 (#读写权限)

此行代码可以将当前对数据库操作的权限设为“读写权限”。





(7) “开始事务 ()”、“保存事务 ()”和“回滚事务 ()”方法

功能: “开始事务 ()”方法, 用来开始一个新事务;

“保存事务 ()”方法, 保存任何更改并结束当前事务;

“回滚事务 ()”方法, 取消当前事务中所作的任何更改并结束事务。

使用事务操作, 相当于对数据库操作过程的备份。“开始事务 ()”后, 会记录下对数据库的操作, 如果使用“回滚事务 ()”, 那么在“开始事务 ()”后对数据库的所有操作都会被取消。可以利用事务处理, 提高数据库操作的安全性。例如, 在对数据库进行一系列操作, 如果其中一个环节出现了错误, 就让数据库恢复到最初的样子, 实例如下:

数据库连接1. 开始事务 ()

如果 (数据库连接1. 执行SQL (“CREATE TABLE 学生信息表3 (姓名 text, 年龄 int, 籍贯 text)”) = 真)

如果 (数据库连接1. 执行SQL (“INSERT INTO 学生信息表 (姓名, 年龄, 籍贯) VALUES (‘张满’, 22, ‘山东’)”) = 真)

信息框 (“操作全部成功”, 0, )

数据库连接1. 保存事务 ()

数据库连接1. 回滚事务 ()

数据库连接1. 回滚事务 ()

在实际应用中, 事务操作是非常重要的, 例如银行系统的数据库操作需要非常高的安全性, 从存储账户到支票账户的转账涉及多个数据库操作, 只有这些数据库操作全部完成, 才完成这次转账。在转账前新建一个事务; 在转账的过程中, 有任何一个环节出现错误, 就回滚事务, 取消前面对数据库的操作; 只有所有操作都成功了, 才“保存事务”。这样大大提高了对数据库操作的安全性。

### 12.3.2 “记录集” 组件

#### 1. “记录集” 组件的重要属性

“记录集”的重要属性有“是否已打开”、“操作状态”、“记录数量”、“字段数量”、“首记录前”、“尾记录后”。

这些属性都是只读属性, 在程序中可以读取属性值, 但不能修改。

“是否已打开”属性, 为逻辑型, 判断数据库表单是否已经打开;

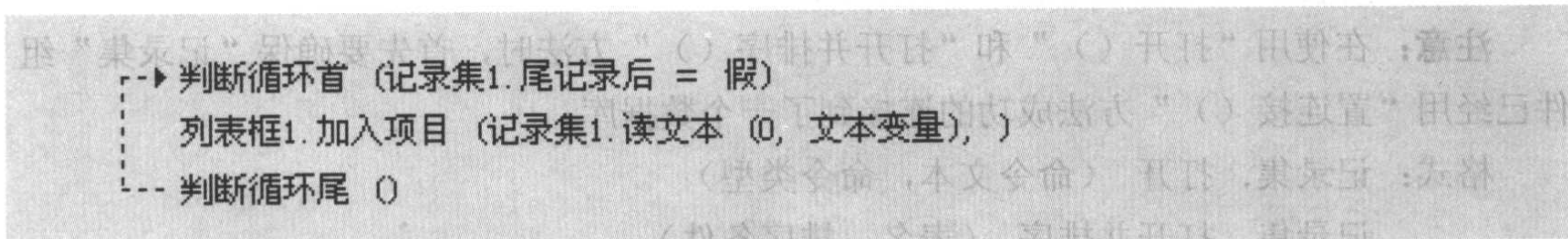
“操作状态”属性, 表示记录集的 17 种操作状态, 具体含义请查看易语言帮助;

“记录数量”和“字段数量”属性, 表示记录集的记录数量和字段数量;

“首记录前”和“尾记录后”属性, 都为逻辑型属性, 判断当前记录指针是否在首记录前或尾记录后。

在程序中, 可以根据需要来读取这些属性的值, 例如, 将“记录集”中第 1 个字段循环加入列表框中, 可以使用“判断循环首 ()”命令, 在循环首进行判断, 如果当前记录指针不在尾记录后, 就进行循环。代码如下:





## 2. “记录集”组件的重要方法

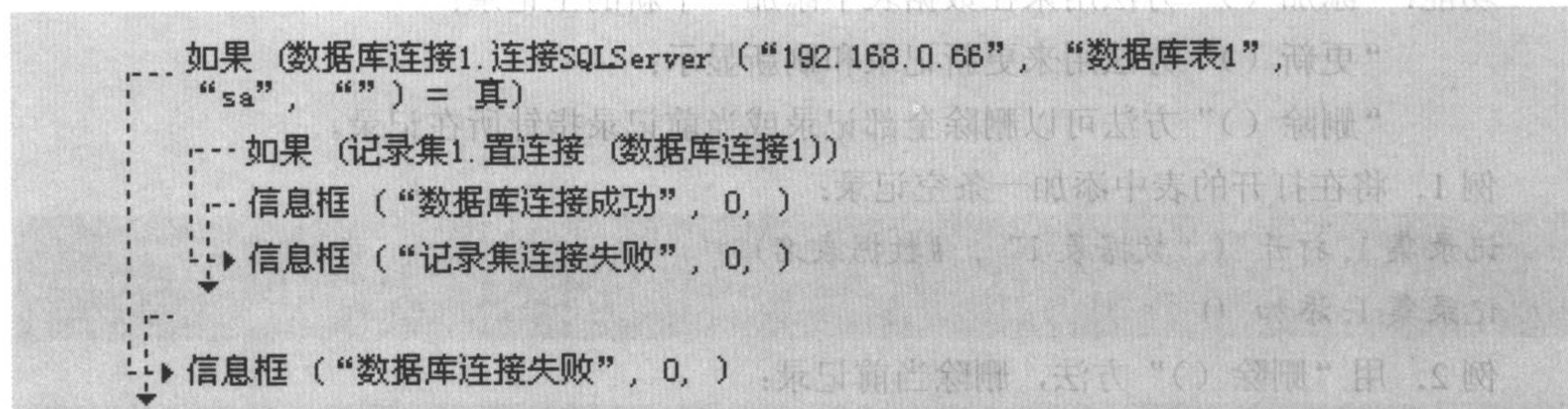
“记录集”的重要方法有“置连接 ()”、“取连接 ()”、“打开 ()”、“打开并排序 ()”、“关闭 ()”、“添加 ()”、“更新 ()”、“删除 ()”、“写文本 ()”、“写双精度 ()”、“写单精度 ()”、“写整数 ()”、“写逻辑值 ()”、“写日期 ()”、“读文本 ()”、“读双精度 ()”、“读单精度 ()”、“读整数 ()”、“读逻辑值 ()”、“读日期 ()”、“保存到 XML ()”、“保存到 ADTG ()”、“到首记录 ()”、“到尾记录 ()”、“到下一条 ()”、“到前一条 ()”、“移到 ()”、“取字段名 ()”、“取字段属性 ()”、“取字段定义长度 ()”、“取字段实际长度 ()”、“取字段类型 ()”、“字段是否为空 ()”、“查找 ()”、“查找下一个 ()”、“过滤记录 ()”。

下面介绍一些常用方法的使用。

### (1) “置连接 ()”方法

功能：用来设置数据库连接，该方法是将“记录集”组件与“数据库连接”组件相连。在参数中直接填入欲相连的“数据库连接”组件的名称。

例如：先将“数据库连接”组件连接到一个数据库，然后将“记录集”组件与数据库关联，使用代码：



### (2) “取连接 ()”方法

功能：取得“记录集”当前的连接，返回值是一个“数据库连接”类型的数据，可以用变量存放该方法的返回值，方便下次连接时使用。

例如，将“取连接 ()”方法的返回值存放在一个“数据库连接”数据类型的变量中：

变量名	类型	静态	数组	备注
数据库连接	数据库连接			

数据库连接 = 记录集1.取连接 ()

### (3) “打开 ()”和“打开并排序 ()”方法

功能：“打开 ()”方法可以用来打开一个表或用来执行一段 SQL 语句。

“打开并排序 ()”方法用来打开数据表，并按照指定条件排序。





**注意：**在使用“打开（）”和“打开并排序（）”方法时，首先要确保“记录集”组件已经用“置连接（）”方法成功的连接到了一个数据库。

格式：记录集. 打开（命令文本，命令类型）

记录集. 打开并排序（表名，排序条件）

例 1，用“打开（）”方法打开一个表：

记录集 1. 打开（“数据表 1”，#数据表名）

例 2，用“打开（）”方法执行一段 SQL 语句：

记录集 1. 打开（“SELECT \* FROM 数据表 1 WHERE 年龄=12”，#SQL 语句）

例 3，用“打开并排序（）”方法，打开一个表并按照表中的“年龄”字段的升顺排列记录：

记录集 1. 打开并排序（“数据表 1”，“年龄 ASC”）

(4) “关闭（）”方法

功能：关闭当前打开的数据表，一般情况下，打开一个数据表后，若不再对表进行操作必须关闭表。

例如，将当前打开数据表 1 关闭：

记录集 1. 关闭（）

(5) “添加（）”、“更新（）”和“删除（）”方法

功能：“添加（）”方法用来在数据表中添加一个新的空记录；

“更新（）”方法用来更新记录和刷新显示；

“删除（）”方法可以删除全部记录或当前记录指针所在记录。

例 1. 将在打开的表中添加一条空记录：

记录集 1. 打开（“数据表 1”，#数据表名）

记录集 1. 添加（）

例 2. 用“删除（）”方法，删除当前记录：

记录集 1. 删除（#删除当前记录）

如何移动当前记录指针将在后面介绍。

(6) “写文本（）”、“写双精度（）”、“写单精度（）”、“写整数（）”、“写逻辑值（）”、“写日期（）”方法

功能：这些方法都用来更改当前记录的指定字段内容，要根据字段的数据类型来选择写的方法，即文本型字段就要用“写文本（）”方法，整数型字段用“写整数（）”方法。

例如，将当前记录的“姓名”字段改写：

记录集 1. 写文本（“姓名”，“王雨”）

(7) “读文本（）”、“读双精度（）”、“读单精度（）”、“读整数（）”、“读逻辑值（）”、“读日期（）”方法



功能：用来读取当前记录指定字段的内容，和写的方法相同，要根据字段类型来选择读的方法，即文本型字段用“读文本（）”方法等。读出来的内容存放在一个变量中，变量的类型也要与读字段的数据类型相符。

例如，读取当前记录的“姓名”字段内容：

局部变量：文本变量      数据类型：文本型

记录集 1. 读文本（“姓名”，文本变量）

(8) “到首记录（）”、“到尾记录（）”、“到下一条（）”、“到前一条（）”、“移到（）”方法

功能：这些方法都用来控制当前记录指针。“移到（）”方法，可以将当前记录指针移动到指定记录。

例 1. 将当前记录指针跳到下一记录位置：

记录集 1. 到下一条（）

例 2. 将当前记录指针移动到第 10 个记录处：

记录集 1. 移到（10）

(9) “取字段名（）”、“取字段属性（）”、“取字段定义长度（）”、“取字段实际长度（）”、“取字段类型（）”、“字段是否为空（）”方法

功能：这些方法用来读取字段的相关信息，可以在程序中根据需要读取这些内容。

例如，在程序中先判断字段的类型，然后再读取记录：

```

--- 如果真 (记录集1.取字段类型 (“性别”) = #逻辑型字段)
    记录集1.读逻辑值 (“性别”，逻辑变量)
    
```

(10) “查找（）”和“查找下一个（）”方法

功能：“查找（）”方法查找满足条件的记录，如果找到记录，将记录指针停留在满足条件的第一个记录上；

“查找下一个（）”方法，查找下一个满足条件的记录。此命令用于“查找（）”命令之后使用，可继续按“查找（）”命令找到下一个满足条件的记录。

格式：记录集. 查找（查找条件，查找方向）

例如，当前已经打开了一个表，在表中找到所有符合“年龄=22”条件的记录，并显示在列表框中，代码如下：

变量名	类型	静态	数组	备注
文本变量	文本型			

记录集1. 到首记录（）

```

--- 如果真 (记录集1. 查找 (“年龄=22”，#正向搜索))
    
```

```

        循环判断首（）
    
```

```

            记录集1. 读文本 (“姓名”，文本变量)
    
```

```

            列表框1. 加入项目 (文本变量, )
    
```

```

        循环判断尾 (记录集1. 查找下一个（） = 真)
    
```





## 12.4 外部数据库应用

### 12.4.1 外部数据库操作例程

下面是演示如何在易语言中借助“外部数据库”组件，通过 ODBC 操纵 Microsoft Access 数据库。支持记录的查询、排序、删除、添加、更新，创建、删除表的操作。本节例程同时为用户提供了良好的 SQL 语言的学习环境。

例程见随书光盘中的“全面操作 Access 数据库.e”和“MSAccess.mdb”文件。

#### 1. ODBC 配置

(1) 运行例程“全面操作 Access 数据库.e”，运行效果如图 12-6 所示。

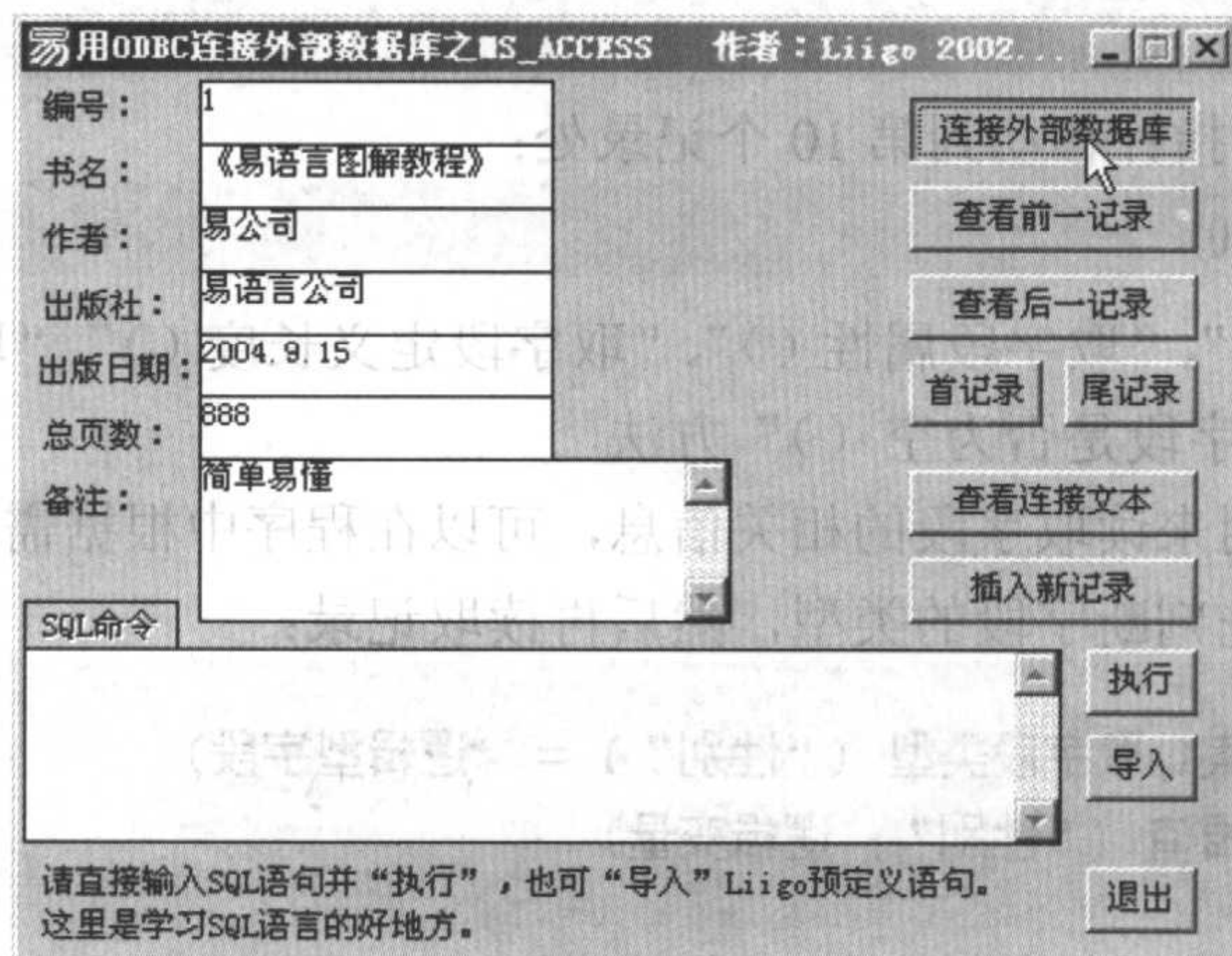


图 12-6 外部数据库操作例程界面

点击“连接外部数据库”按钮，会弹出“选择数据源”对话框。如图 12-7 所示。

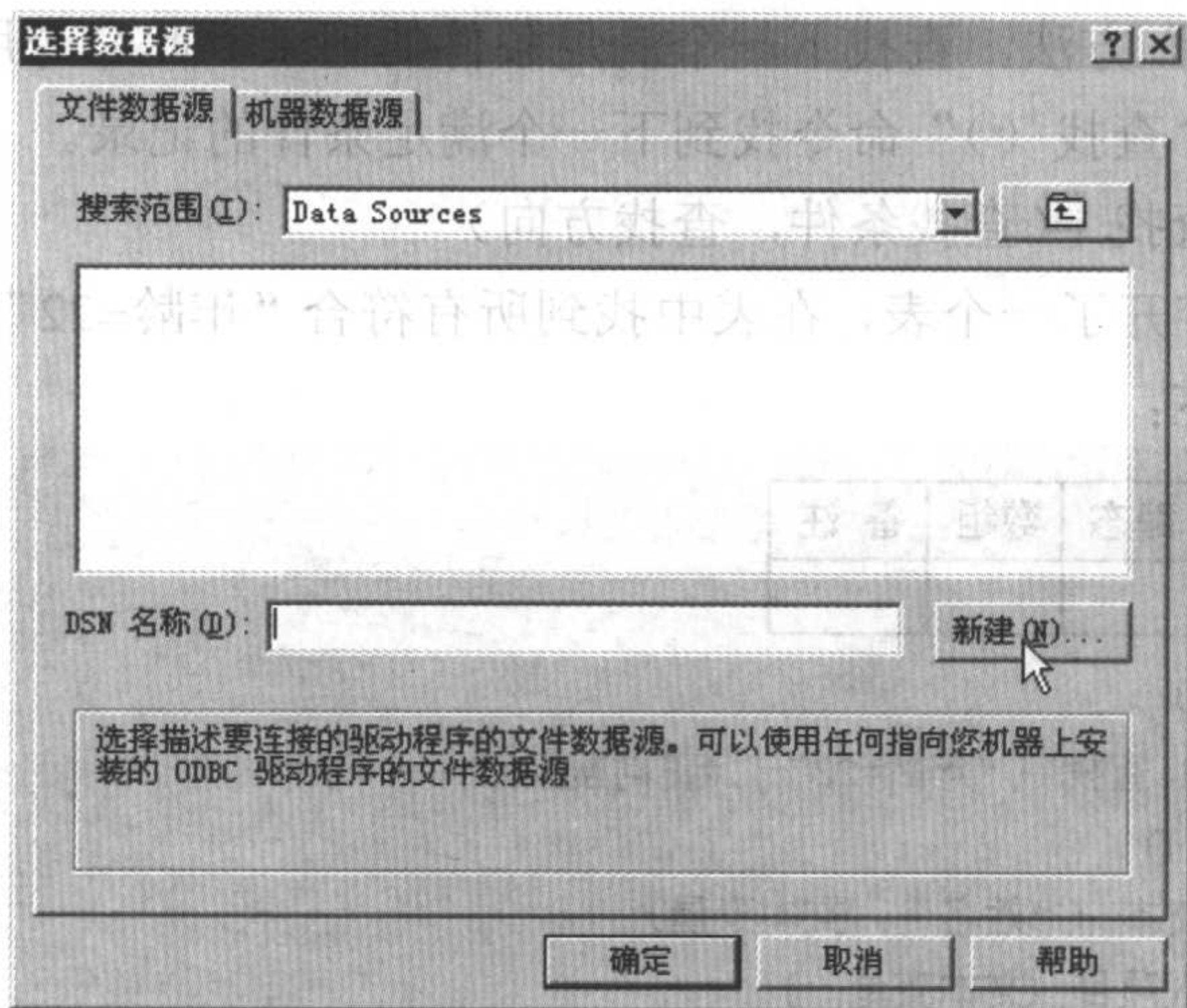


图 12-7 选择数据源对话框



(2) 选择图 12-7 中的“新建”按钮，又弹出一个“创建数据源”对话框，如图 12-8 所示。



图 12-8 创建数据源对话框

(3) 在图 12-8 中的对话框，选中“Driver do Microsoft Access”（Access 的驱动程序），然后单击“下一步”按钮。将再次弹出一个对话框，如图 12-9 所示，要求输入一个带路径的文件名。

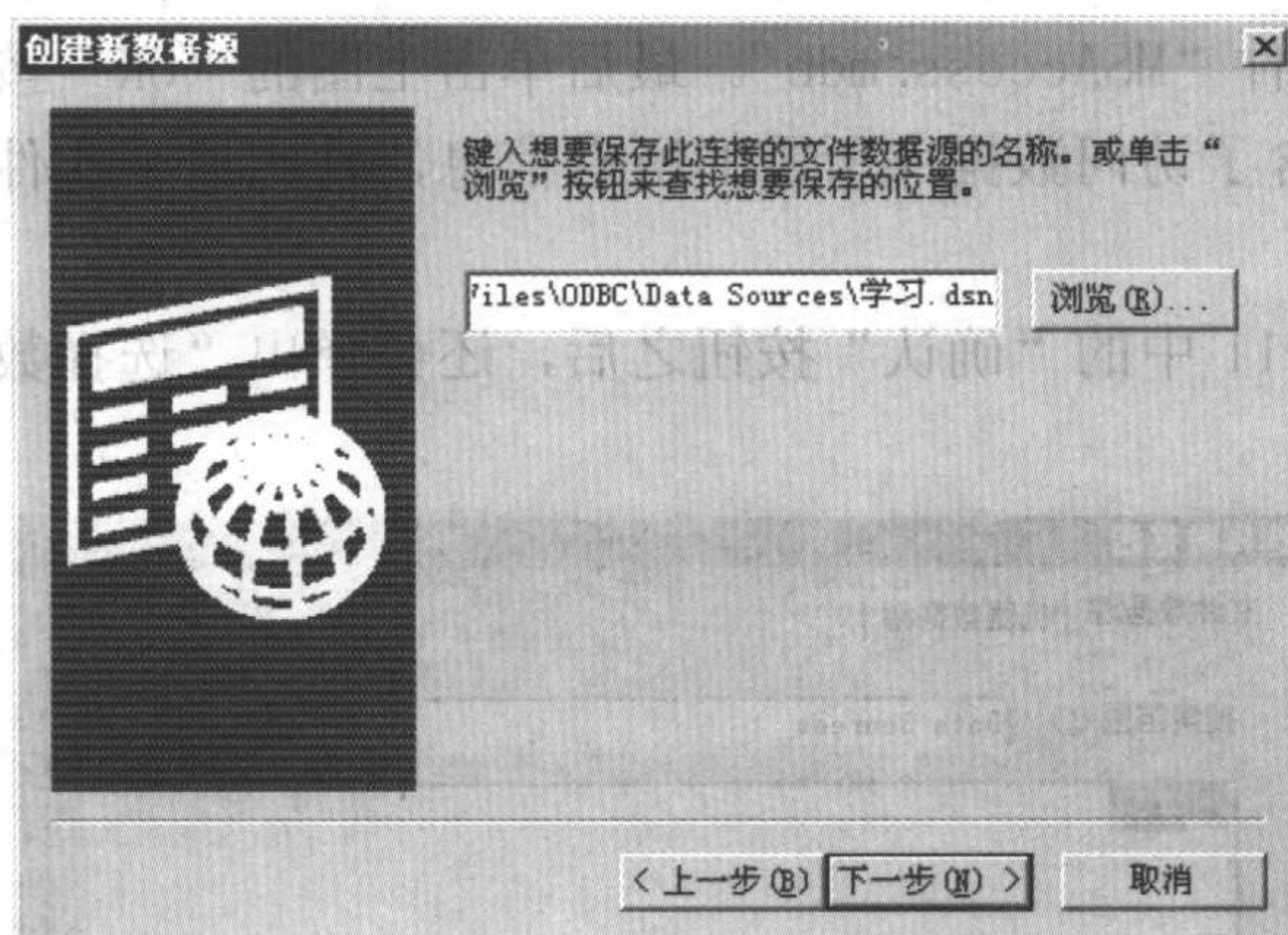


图 12-9 输入保存位置

(4) 点击图 12-9 中的“浏览”按钮，在弹出的“另存为文件对话框”中任选一个路径，填入“学习.dsn”，点“保存”关闭当前窗口。

点击“下一步”按钮后，会弹出确认的对话框，列出了前几步所设置的信息。如果确认，点击“完成”，否则点击“上一步”修改。如图 12-10 所示。



图 12-10 完成创建





(5) 前面几步大家新建了一个“学习.dsn”文件（已设置了路径和驱动程序信息，但还没有和具体的数据库相连接）。在单击“完成”按钮后，会弹出如图 12-11 所示的对话框。可在这个对话框中连接数据库。

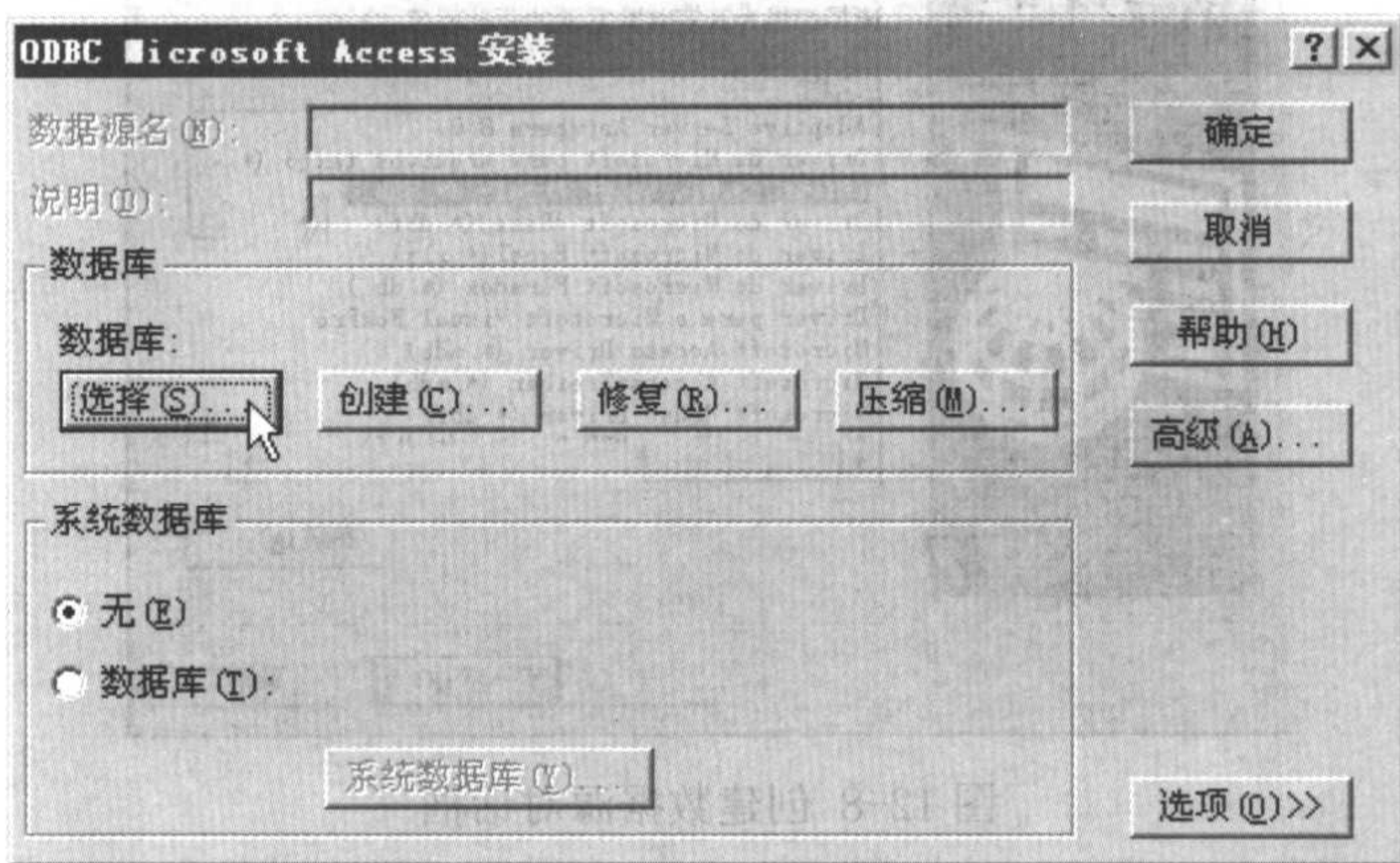


图 12-11 连接数据库

(6) 单击图 12-11 中的“选择”按钮，在弹出的“打开文件对话框”中选择书中所提供的 Access 数据库文件“MSAccess.mdb”。最后单击上图的“OK”按钮。这样新建的数据源连接 DSN 文件就包含了访问数据库所需的所有信息，使用这个文件可以在易语言中操作该数据库了。

(7) 在单击图 12-11 中的“确认”按钮之后，还会弹出“选择数据源”对话框，如图 12-12 所示。

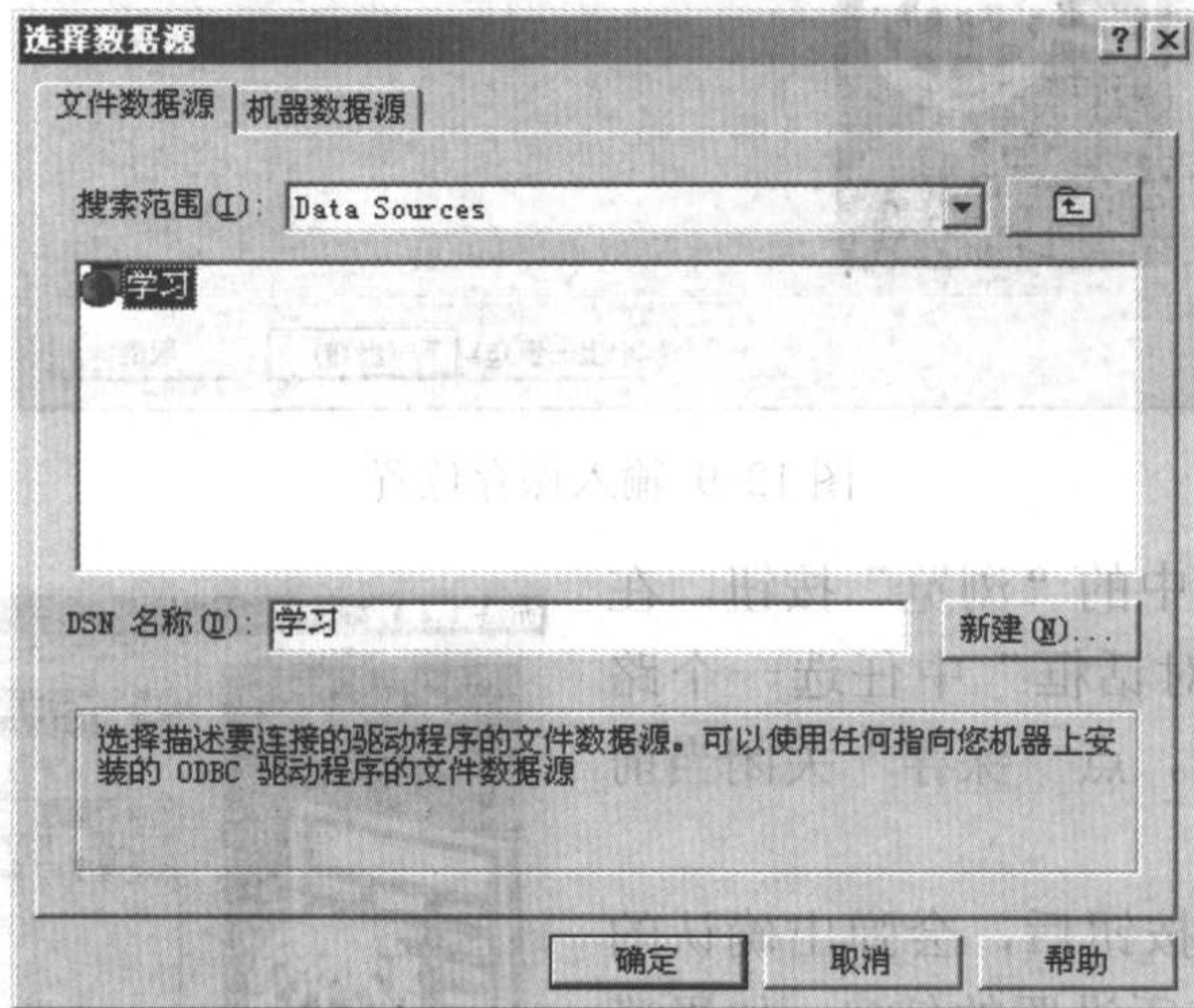


图 12-12 选择“.dsn”文件

可以发现，上图跟前面第一张图是同一个对话框，只是本图所示对话框的中间区域列出了一个文件图标，这就是先前所新建的文件“学习.dsn”，选中它，点击“确定”。这时又会弹出“Access 安装”对话框，如图 12-13 所示。



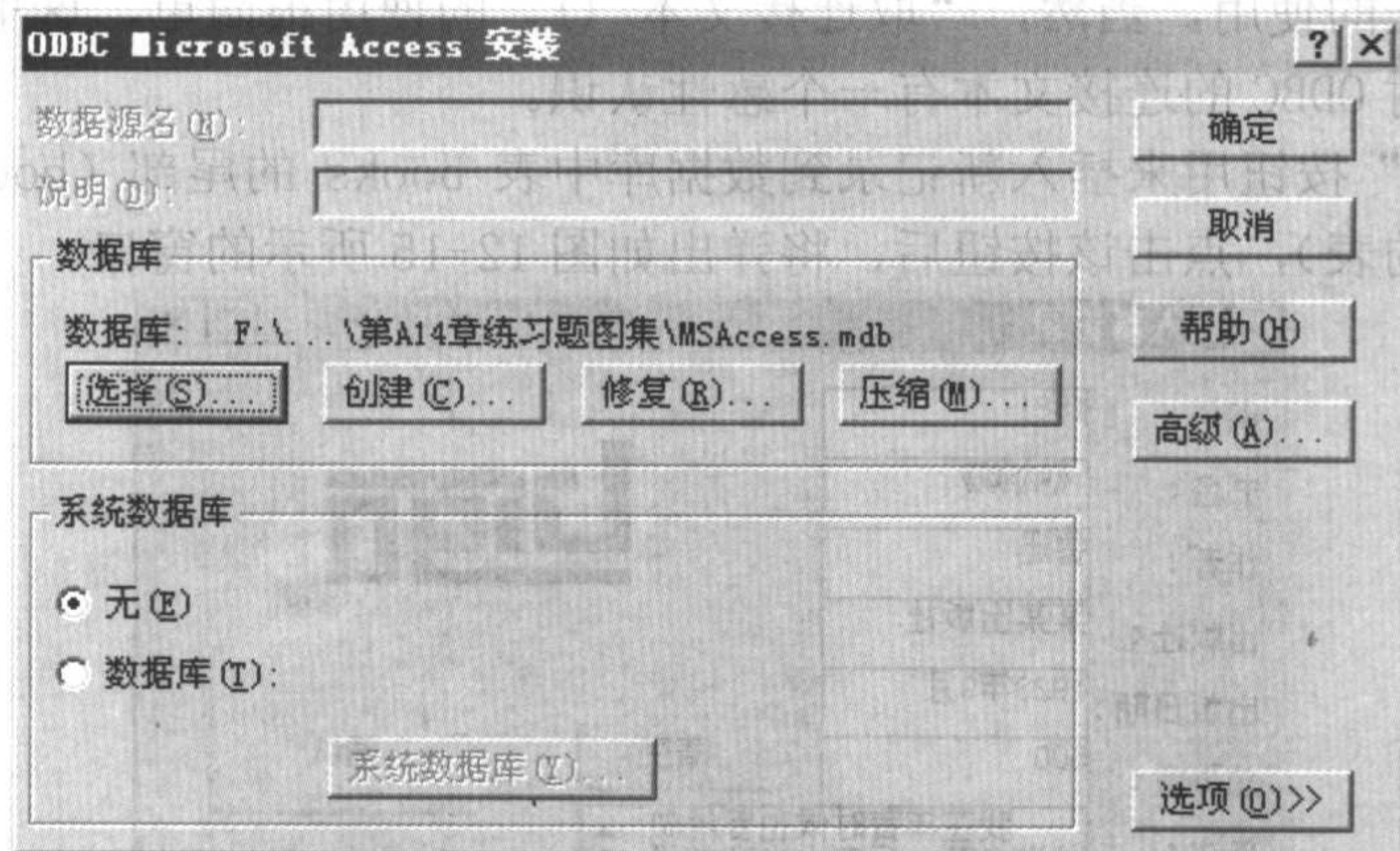


图 12-13 连接数据库

图 12-13 与图 12-11 有些类似，只是中间多了一行数据库的路径名，这正是所要连接的数据库。单击“确定”，就完成了连接！

至此，用 ODBC 连接外部数据库的操作全部完成。大家也许感觉有些复杂，不过这是 ODBC 配置的标准过程。值得一提的是，下次再连接同一数据库时，就不必再新建一个\*.dsn 文件了，直接在最前面的对话框（图 12-12）中选择“学习.dsn”就行了。

关闭图 12-13 所示的对话框后，程序又回到了“全面操作 Access 数据库.e”主界面，并且已经正确地读出并显示了 Access 数据库的第一条记录。如图 12-14 所示。

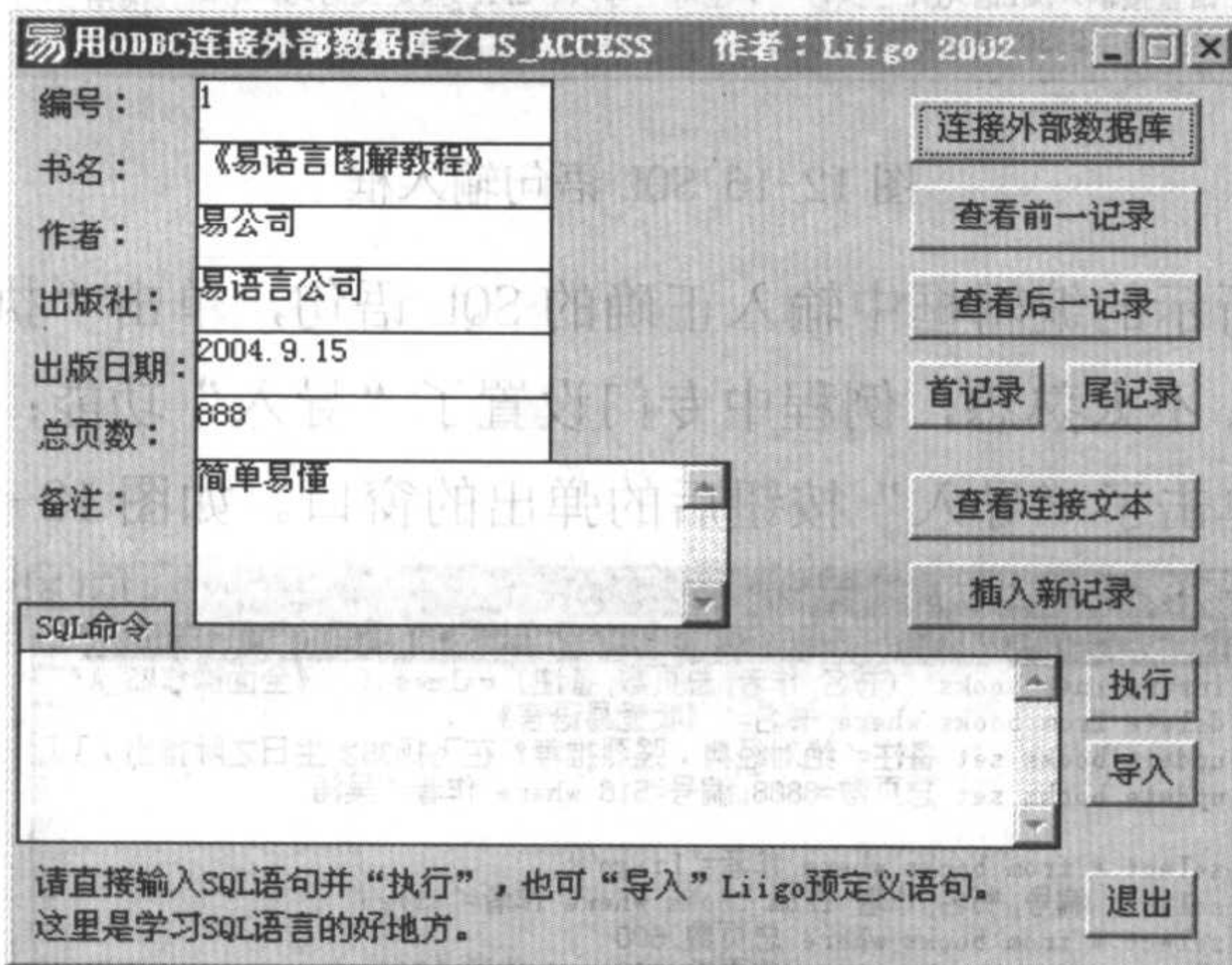


图 12-14 数据库被打开

## 2. 关于例程的使用

上面的例程参见随书光盘例程“全面操作 Access 数据库.e”。

例程中的“连接外部数据库”按钮前面已说的很详细了。

下面的四个按钮“查看前一记录”、“查看后一记录”、“首记录”、“尾记录”是用于浏览数据库中的记录的，大家可以双击按钮查看其中的程序代码。

因为“外部数据库”组件有个“取连接文本 ()”的方法，故专门提供按钮“查看连接





文本”演示本方法的使用，当然，“取连接文本 ()”的使用很简单，这里提供它的目的，只是为了让大家对 ODBC 的连接文本有一个感性认识。

“插入新记录”按钮用来插入新记录到数据库中表 books 的尾部 (books 是在 Access 数据库中预定义的表)，点击该按钮后，将弹出如图 12-15 所示的窗口。

易语言插入新记录

编号: 1881

书名: 《呐喊》

作者: 鲁迅

出版社: 某某出版社

出版日期: 1923年9月

总页数: 500

备注: 我在年青时候也曾经做过许多梦, 后来大半忘却了, 但自己也并不以为可惜。所谓回忆者, 虽说可以使人

清空 确认 关闭

图 12-15 插入新记录

填好后单击“确认”按钮就可以插入新记录了。

在主界面的下部，是专门为大家订做的一个用于练习使用 SQL 语言的环境。

SQL命令

执行 导入 退出

请直接输入SQL语句并“执行”，也可“导入”Liigo预定义语句。  
这里是学习SQL语言的好地方。

图 12-16 SQL 语句输入框

只要在图 12-16 所示的编辑框中输入正确的 SQL 语句，单击“执行”按钮就可以了。考虑到大家可能对 SQL 不太熟悉，例程中专门设置了“导入”功能：即可以从预定义的语句中选取。下图是在单击了“导入”按钮后的弹出的窗口。如图 12-17 所示。

易语言预定义SQL语句

insert into books (编号, 书名, 作者, 出版社, 出版日期, 总页数, 备注) values ('1881', '《呐喊》', '鲁迅', '某某出版社', '1923年9月', '500', '我在年青时候也曾经做过许多梦, 后来大半忘却了, 但自己也并不以为可惜。所谓回忆者, 虽说可以使人')

insert into books (书名, 作者, 总页数, 备注) values ('《全面操作MS Access》', 'Liigo', '800', '绝对经典, 强烈推荐! 在飞扬35岁生日之际推出, 强烈推荐!')

delete from books where 书名='《我爱易语言》'

update books set 备注='绝对经典, 强烈推荐! 在飞扬35岁生日之际推出, 强烈推荐!'

update books set 总页数=8888, 编号=518 where 作者='吴涛'

select \* from books where 作者='Liigo'

select 编号, 书名, 作者 from books where 作者='Liigo'

select \* from books where 总页数>800

select \* from books where 总页数>800 and 作者='liigo'

select \* from books order by 总页数

select \* from books order by 总页数 desc

create table e lovers (编号 integer, 姓名 varchar(10), 邮箱 varchar(20))

insert into e lovers (姓名, 邮箱, qq) values ('西风', 'westwindxp@hotmail.com', '123456')

insert into e lovers (姓名) values ('liigo')

delete from e lovers

drop table e lovers

以上各语句都已经过严格测试，至少不会有语法错误。liigo@chinaren.com  
双击选择。右击可查看帮助。 Liigo, Liigo@chinaren.com

图 12-17 导入 SQL 语句

双击鼠标可导入选定的 SQL 语句，单击鼠标右键可查看各 SQL 语句的功能。可以看到，



在这些预定义语句中，包含了对记录的查询、排序、添加、删除、更新，及其他对“表”的操作。当然，这些语句都是最基本的 SQL 语句。

### 3. 易语言中使用 SQL 语句的注意事项

以本程序中用到的这样一段语句为例：

-启动窗口.外部数据库 1.执行 ("insert into books " + "(编号,书名,作者,出版社,出版日期,总页数,备注)" + " values " + "(" + 编辑框 1.内容 + "," + 编辑框 2.内容 + "," + 编辑框 3.内容 + "," + 编辑框 4.内容 + "," + 编辑框 5.内容 + "," + 编辑框 6.内容 + "," + 编辑框 7.内容 + "'))"

注意：

- 关键字要写对，如不要把 values 写为 value，(大小写无所谓)；
- values 与前后的括号之间要有空格（半角空格，不可以是全角）；
- values 前的括号中列出预插入的字段名称，之间以半角空格隔开(不可以是全角)；
- values 后的括号中列出预插入的字段的价值，字符串要以半角单引号或[ ]括住。
- values 之前的括号中的字段名称，与之后的括号中的字段的值，要一一对应。

### 4. 总结

本例程演示了如何在易语言中应用“外部数据库”组件，通过 ODBC 操纵 Microsoft Access 数据库。支持记录的查询、排序、删除、添加、更新，及创建、删除表的操作。本程序同时为用户提供了一个良好的 SQL 语句的学习环境。

### 5. 另一种操作方法

通过新增的外部数据库提供者及数据源连接快速显示记录。

#### (1) 增加两个组件

增加外部数据库提供者组件与数据源组件。如图 12-18 所示。

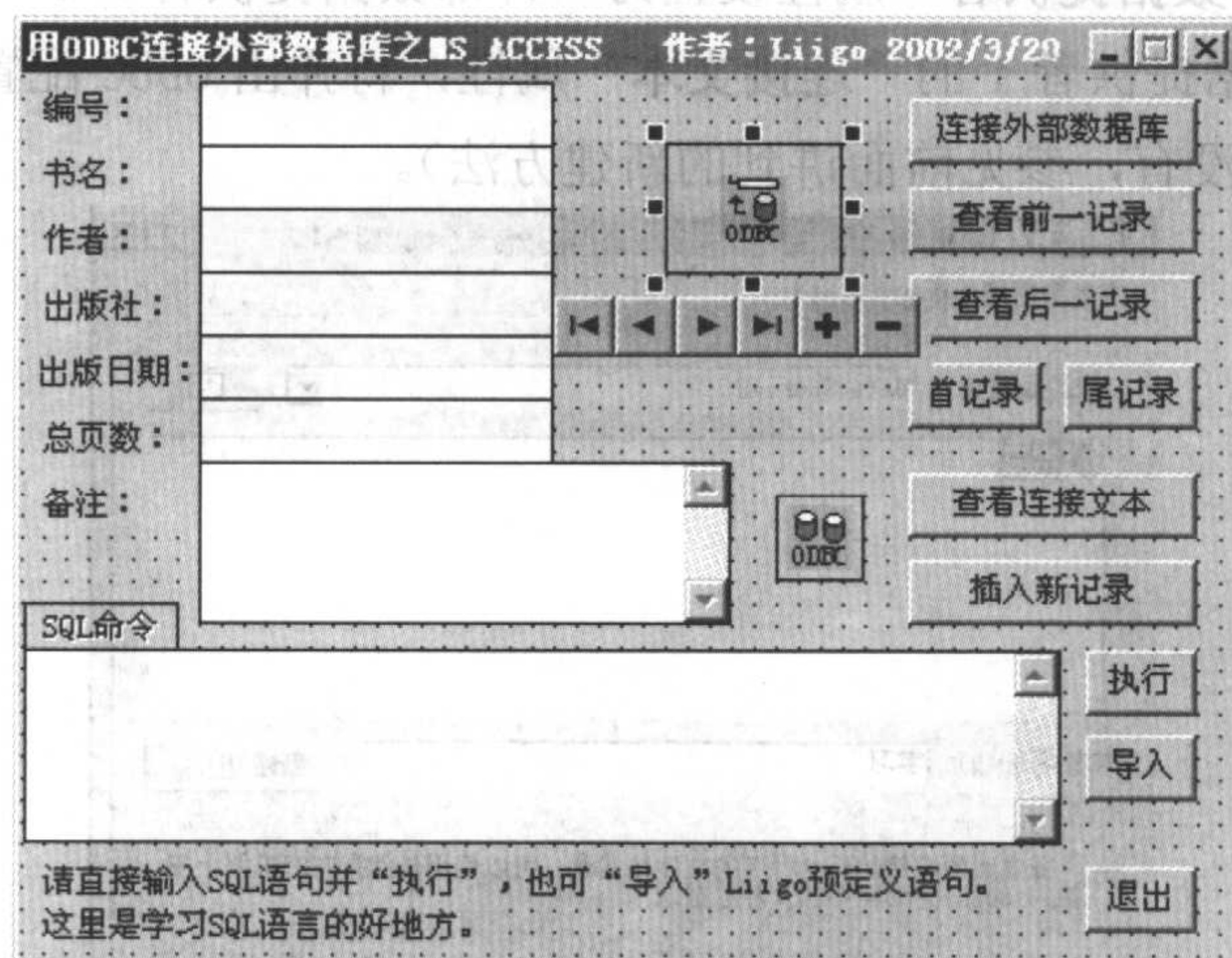


图 12-18 设计“全面操作 Access 数据库 2.e”例程

其中要求将外部数据库提供者组件的“连接文本”属性选为数据库。可使用本书提供的例库。





将外部数据库提供者组件的“查询 SQL”属性选为此数据库的表。

## (2) 改编框的属性

将所有编辑框的“数据源”属性改为“数据源 1”。

将所有编辑框的“数据列”属性改为相对应的属性。

## (3) 试运行

试着运行，看看是不是非常快速的显示 Microsoft Access 数据库中的内容了。

本例程文件名为“全面操作 Access 数据库 2.e”。

## 12.4.2 用表格组件显示数据库

下面为大家介绍一下如何用表格组件显示 Microsoft Access 数据库。

1. 新建一个易程序，在“\_启动窗口”中添加表格、数据源、外部数据提供者组件各一个。如图 12-19 所示。

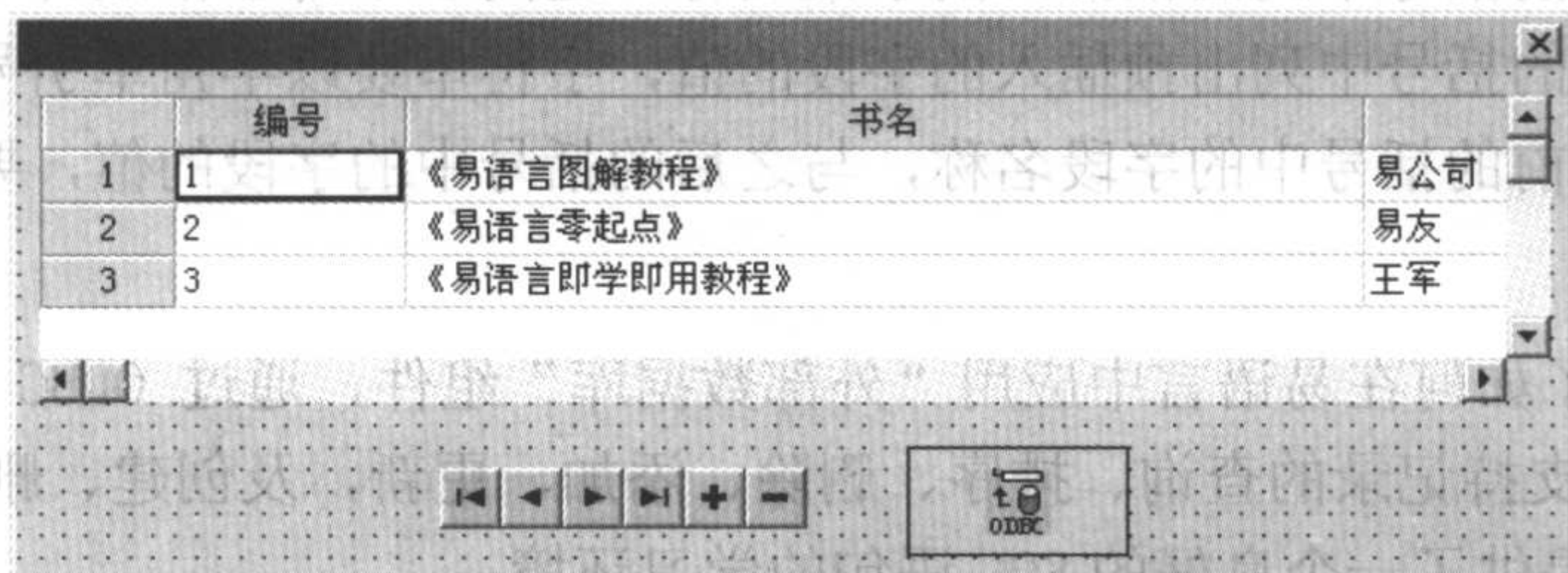


图 12-19 程序布局

## 2. 设置属性

将表格 1 的“数据源”属性设置为“数据源 1”。

将数据源 1 的“数据提供者”属性设置为“外部数据提供者 1”。

3. 点击外部数据提供者 1 的“连接文本”属性，将弹出 ODBC 配置对话框，选择相应的\*.dsn 文件（如果没有，参见前面讲到的新建方法）。

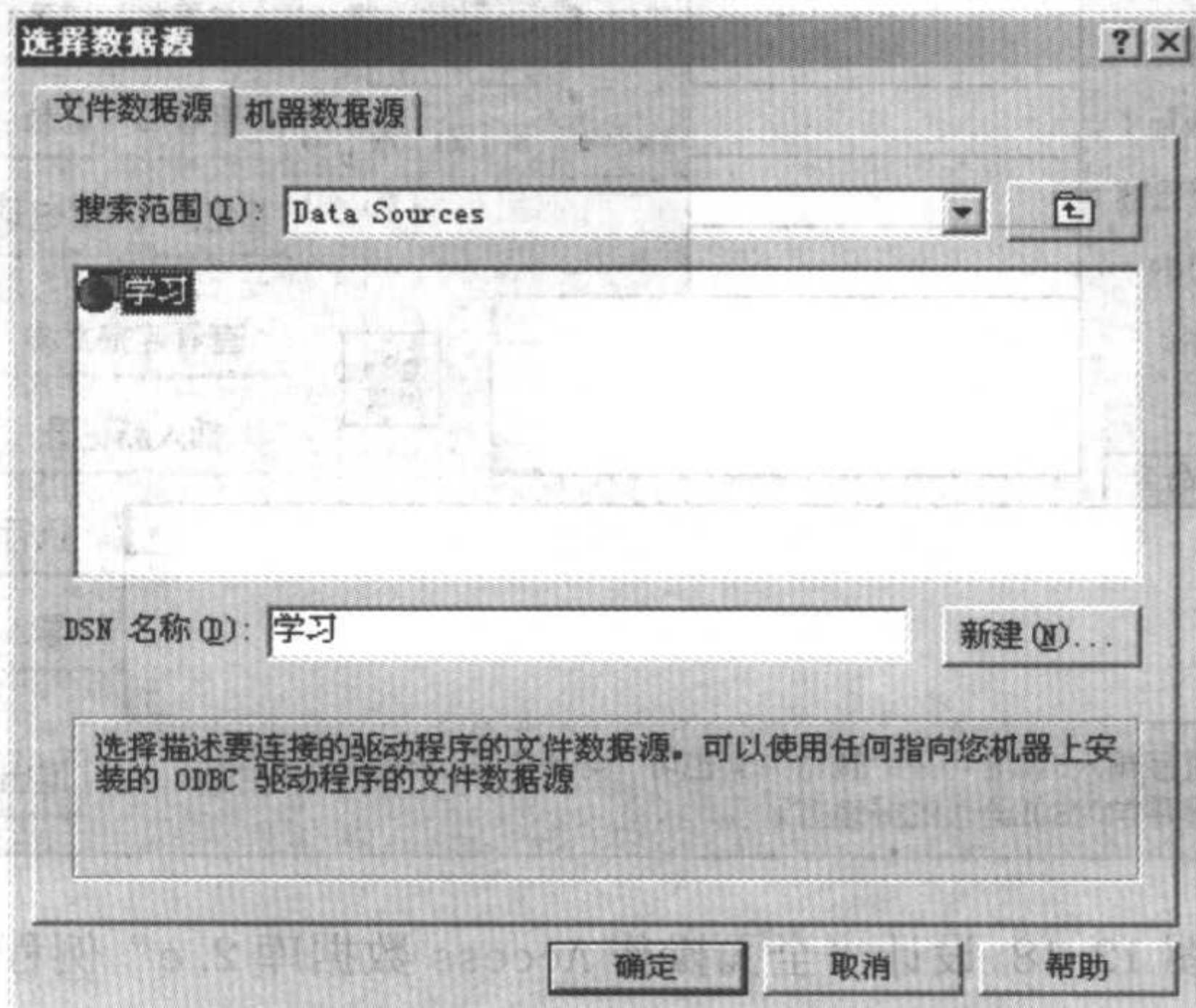


图 12-20 选择 dsn 文件



#### 4. 设置属性

设置外部数据提供者 1 组件的“查询 SQL”属性为“select \* from books”这里 books 是表名，依数据库而定。

#### 5. 运行

这样就完成了用表格组件显示 Microsoft Access 数据库的任务。可以直接用数据源组件的按钮对数据库进行操作，如跳到下一记录、删除记录等等。如图 12-21 所示。

**注意：**数据源的删除记录按钮不可随便点击。因为“数据源”将直接对数据库进行操作，因此删除后的项目将无法恢复。

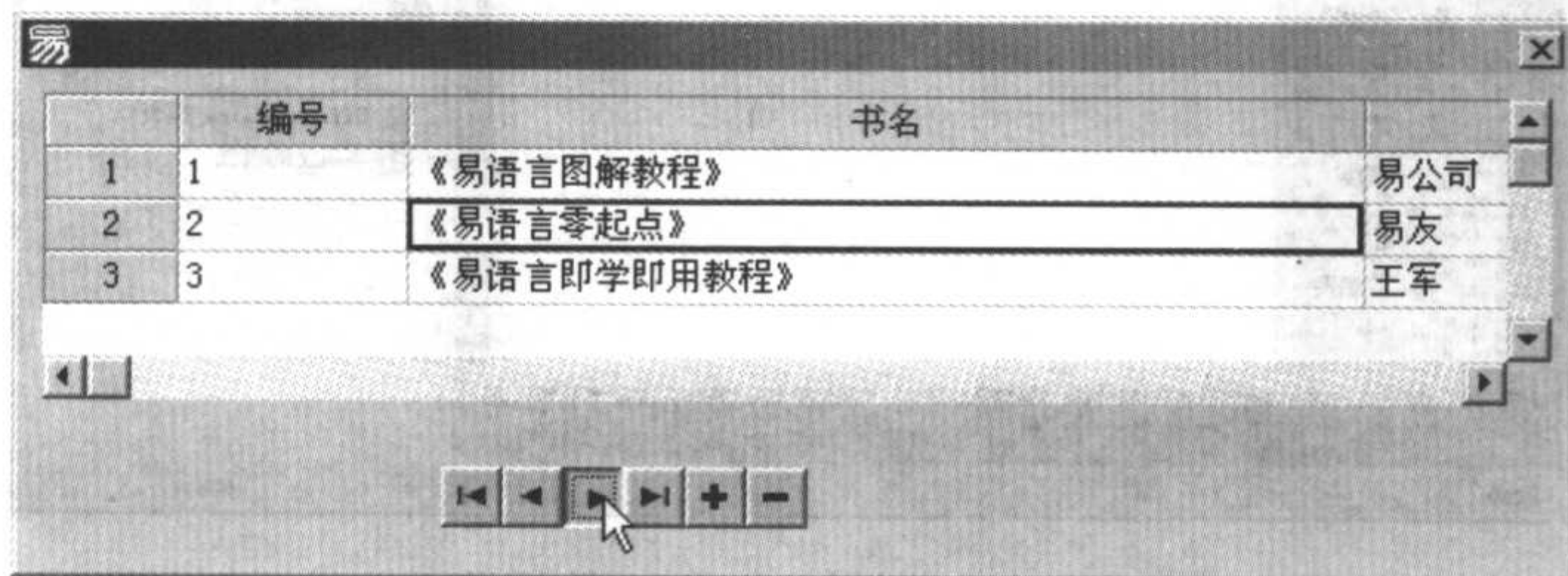


图 12-21 用“数据源”的按钮对数据库进行操作

## 12.5 Access 数据库

### 12.5.1 Access 数据库简介

Access 是微软 Office 软件包中的一个组件，用来建立和管理 Access 数据库。Access 用于按表格的形式来应用数据记录。

跟易数据库不同，Access 数据库内可以有多个表，比如一个学生管理系统数据库内包含学生成绩表、学生基本情况表、管理人员表等多个表。这样就很方便地将相关的字段集中在一起管理，各个表之间又可以相互关联。微软的 Access 数据库软件界面或一个表的界面大约跟易之表相似，但功能更强大。

数据必须以记录的形式存储在记录表中，所以在保存数据之前，必须在数据库中创建相应的数据表。

使用另一种方法也可以实现对其他数据库的访问，这就是使用易语言提供的“数据库格式转换”功能：即易语言安装目录 Tools 目录下的 Dbcnv.exe 程序，可双击执行，也可通过易语言主菜单“数据库”→“数据库格式转换”启动。

### 12.5.2 Access 数据库综合例程

Access 数据库的建立方法可以分成两个阶段，第一个阶段是根据要输入的数据性质，新增表并设置表的字段名称、数据类型和语句；第二个阶段是在表内输入数据。

下面就以建立学生数据库(学生.mdb)中的学生成绩表(chj)为例说明如何建立 Access





数据库。

(1) 打开 Microsoft Access 2000, 然后点击工具条中的“新建”按钮, 在右边的窗口中选择“空数据库...”选项。如图 12-22 所示。

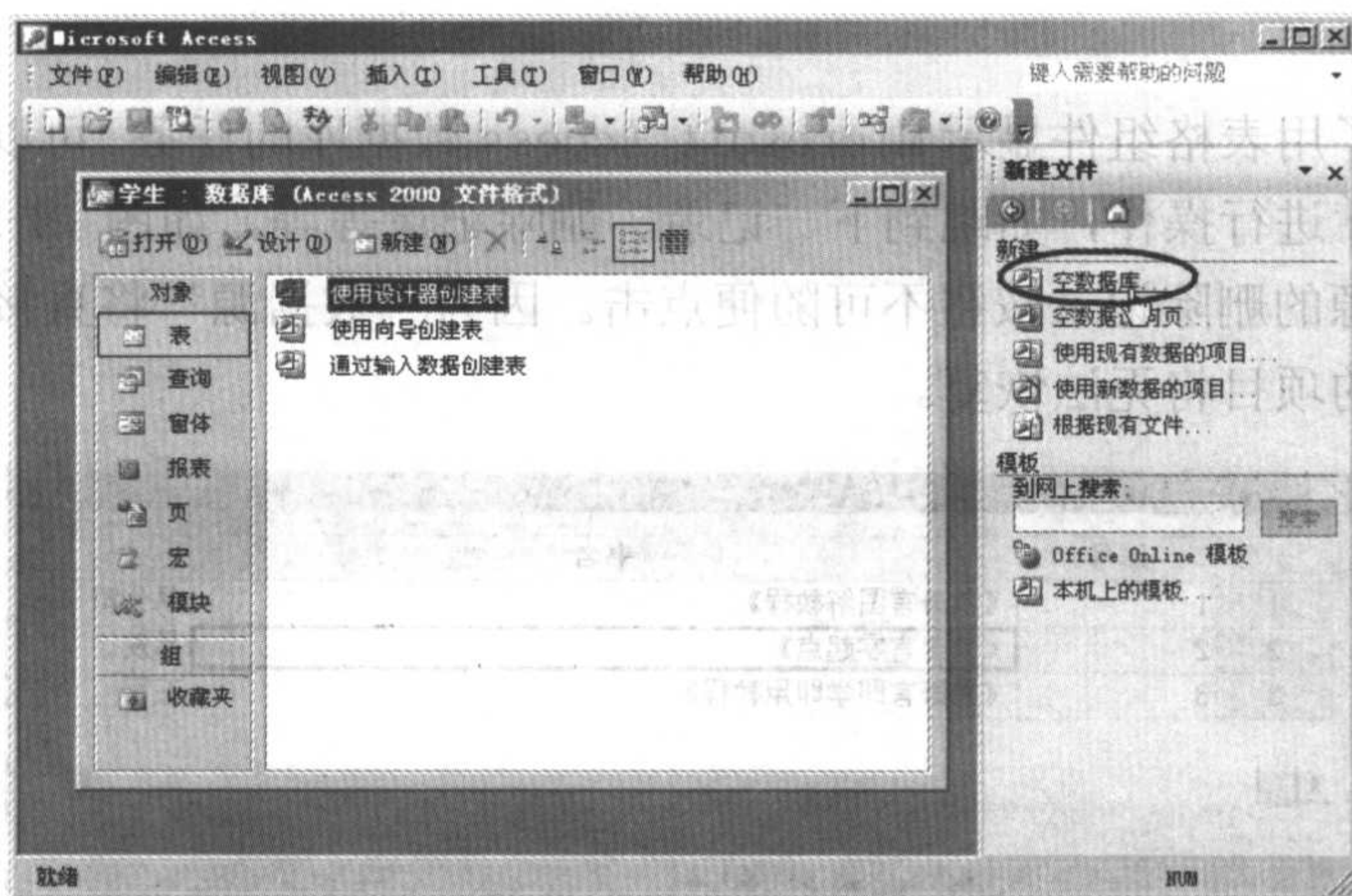


图 12-22 建立 Access 数据库

(2) 弹出“文件新建数据库”对话框, 指定新数据库的储存位置、文件名称(本例为“学生.mdb”), 然后单击“创建”按钮。

(3) 弹出名为“学生数据库”的窗口, 在“使用设计器创建表”上双击鼠标。

(4) 在“字段名称”输入表的第一个字段名称“学号”, 在“数据类型”的下拉菜单中选“数字”, 然后在“说明”中输入“学号”字段的说明文字。

(5) 依此类推, 分别设置“姓名”、“语文”、“数学”、“英语”等字段, 然后关闭表, 此时, Office 助手会询问您是否储存表, 请点取“是”。如图 12-23 所示。

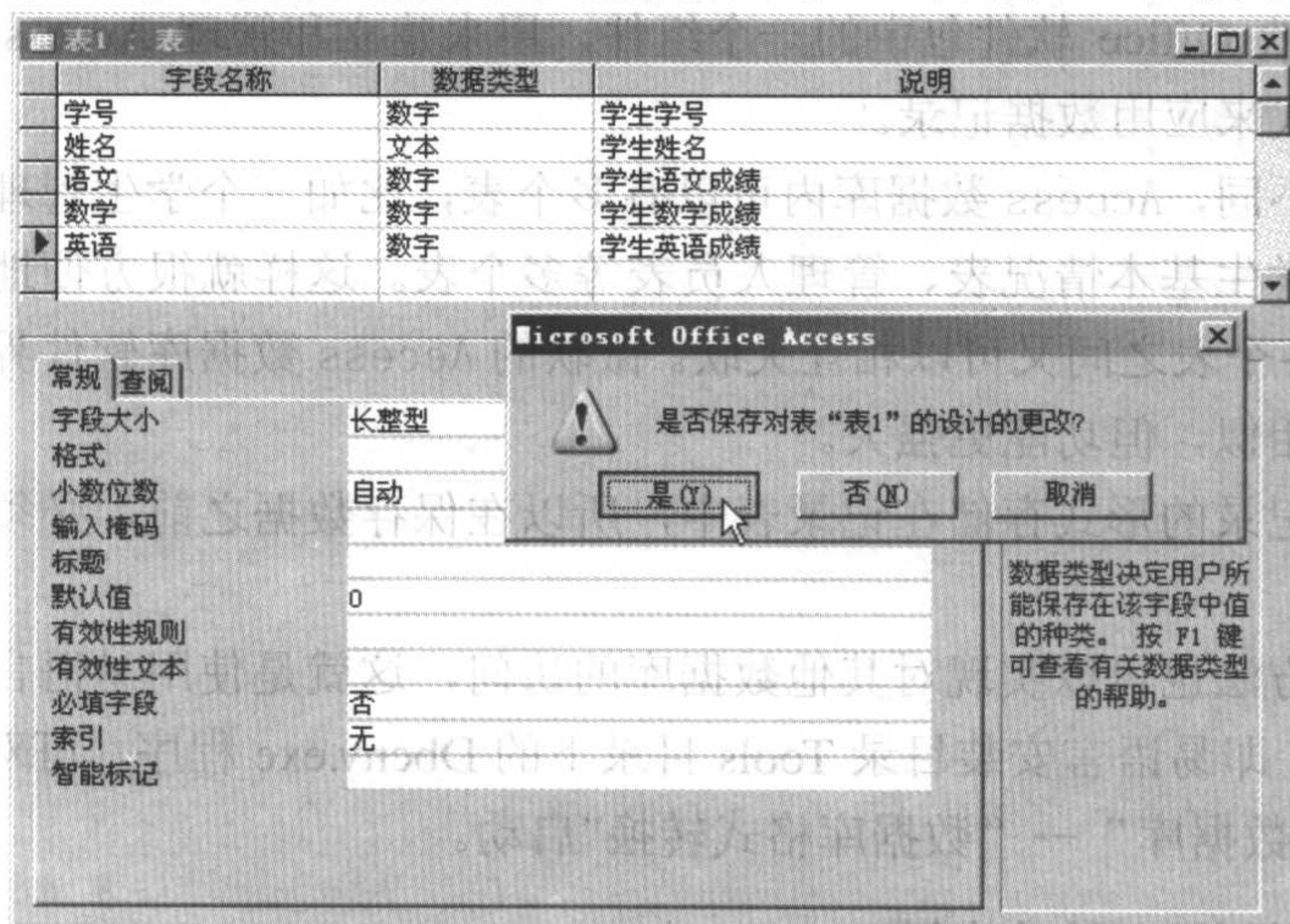


图 12-23 保存新建的表

(6) 在“另存为”对话框的“表名称”框中输入表的名称为“成绩表”, 然后按“确定”。

(7) 在确认是否为数据表创建主键之后, 就完成了数据表设计工作。可以在“学生数



数据库”窗口中看到新增的名为“成绩表”的数据表，如图 12-24 所示。

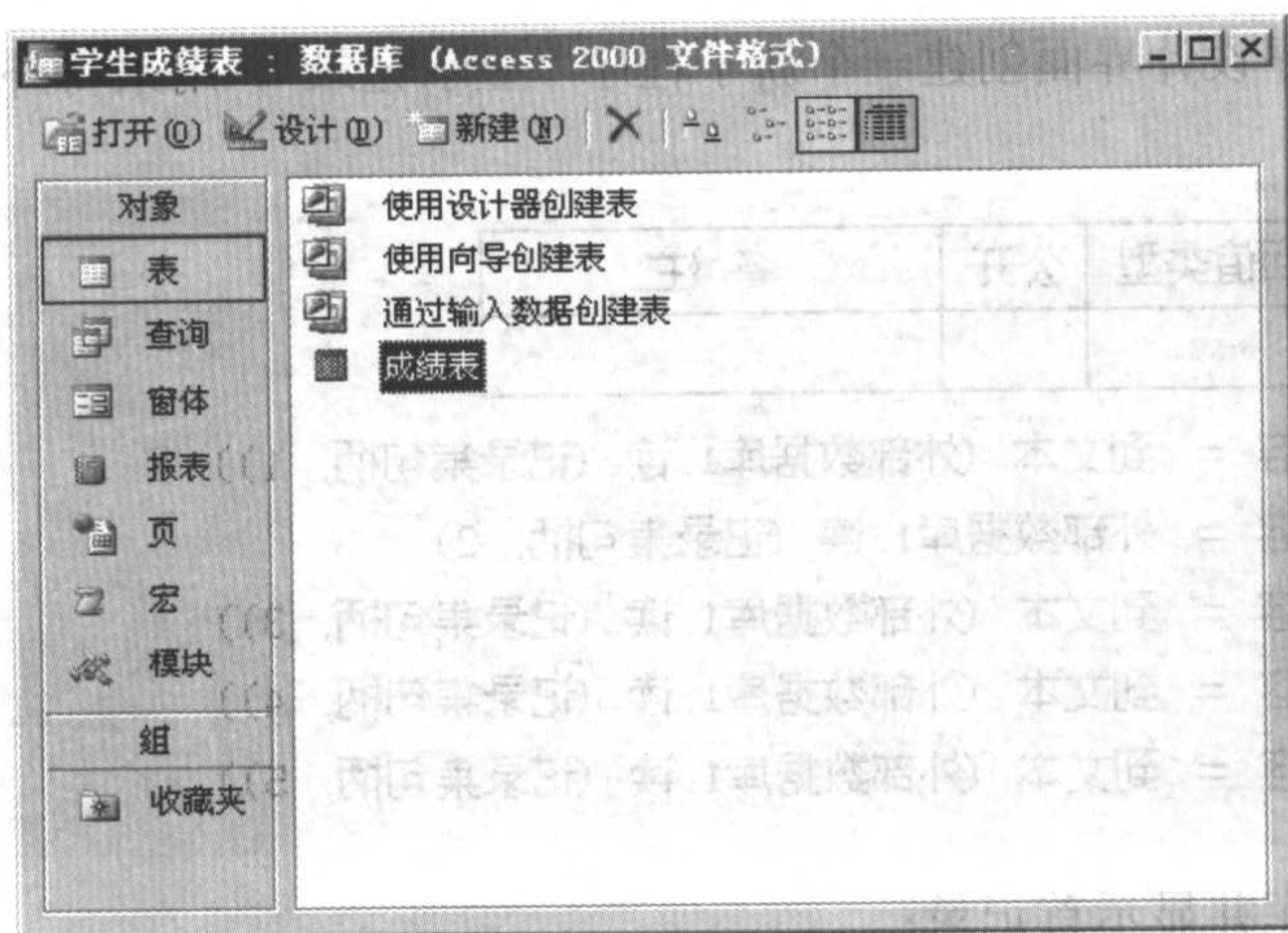


图 12-24 “成绩表”创建完毕

(8) 在“学生数据库”窗口新增的表的名称上双击鼠标，依次输入记录，完毕后可以关闭 Access，Access 会自动存盘。如果需要再建立其他表，可以重复以上 (3) — (8) 的步骤。

前面学习了一些外部数据库的知识，下面就来制作一个简单的学生成绩管理系统，通过它来学习如何实现查询、到首记录、到尾记录、到上一记录、到下一记录等查询动作以及增删、更新等编辑动作。

### 1. 设置数据库及程序界面

(1) 建立一个名为“学生.mdb”的数据库，并新建一个名为“成绩表”的成绩表，加上“学号”、“姓名”、“语文”、“数学”、“英语”五个字段，设置“学号”字段的类型为“自动编号”，并设定为主键。

(2) 在启动窗口中加上名为“学号编辑框”、“姓名编辑框”、“语文编辑框”、“数学编辑框”、“英语编辑框”这五个编辑框作为数据处理者，再加上标题为“到首记录”、“上一记录”、“下一记录”、“到尾记录”、“添加记录”、“查询记录”、“修改记录”、“删除记录”这八个按钮，添加 5 个标签组件用于提示，添加 1 个“外部数据库”组件。如图 12-25 所示。

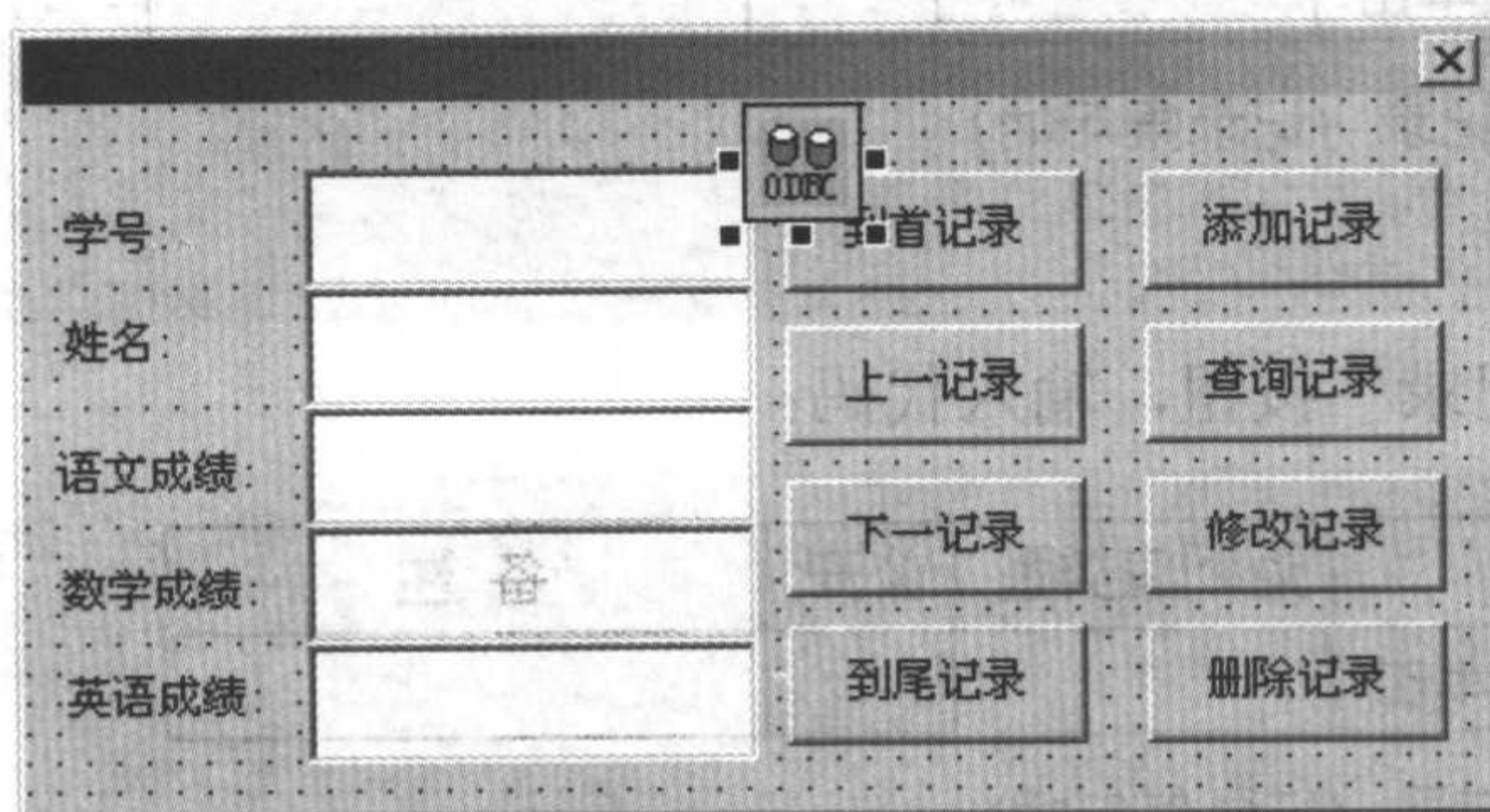


图 12-25 学生成绩管理界面

(3) 将学号编辑框的“输入方式”属性设为“只读方式”。





## 2. 编写实现程序功能的各子程序代码

(1) 首先在程序设计界面创建一个新子程序“显示记录”，用来将外部数据库中的记录显示在编辑框中：

子程序名	返回值类型	公开	备注
显示记录			

学号编辑框.内容 = 到文本 (外部数据库1.读 (记录集句柄, 1))

姓名编辑框.内容 = 外部数据库1.读 (记录集句柄, 2)

语文编辑框.内容 = 到文本 (外部数据库1.读 (记录集句柄, 3))

数学编辑框.内容 = 到文本 (外部数据库1.读 (记录集句柄, 4))

英语编辑框.内容 = 到文本 (外部数据库1.读 (记录集句柄, 5))

## (2) 打开数据库并显示首记录：

在“\_\_启动窗口\_创建完毕”子程序中输入代码：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

```
如果真 (外部数据库1.打开 ("ODBC;DBQ=E:\教材\学生.mdb;DefaultDir=E:\教材;Driver={Driver do Microsoft Access (*.mdb)};DriverId=25;FIL=MS Access;FILEDSN=D:\Program Files\Common Files\ODBC\Data Sources\学习.dsn;MaxBufferSize=2048;MaxScanRows=8;PageTimeout=5;SafeTransactions=0;Threads=3;UID=admin;UserCommitSync=Yes;" , ))
```

```
记录集句柄 = 外部数据库1.查询 ("select * from 成绩表 ")
```

```
如果真 (记录集句柄 = 0)
```

```
信息框 ("查询失败!", 0, )
```

```
外部数据库1.到首记录 (记录集句柄)
```

```
显示记录 ()
```

```
姓名编辑框.获取焦点 ()
```

## (3) 双击“到首记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_到首记录按钮_被单击			

外部数据库1.到首记录 (记录集句柄)

显示记录 ()

## (4) 双击“到尾记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_到尾记录按钮_被单击			

外部数据库1.到尾记录 (记录集句柄)

显示记录 ()



(5) 双击“到上一记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_向前按钮_被单击			

外部数据库1.到前一记录 (记录集句柄)

--- 如果 (外部数据库1.首记录前 (记录集句柄))

信息框 (“前面已无记录!”, 0, )

--- 外部数据库1.到首记录 (记录集句柄)

▶ 显示记录 ()

(6) 双击“到下一记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_向后按钮_被单击			

外部数据库1.到后一记录 (记录集句柄)

--- 如果 (外部数据库1.尾记录后 (记录集句柄))

信息框 (“后面已无记录!”, 0, )

--- 外部数据库1.到尾记录 (记录集句柄)

▶ 显示记录 ()

(7) 双击“添加记录”按钮，在“\_添加按钮\_被单击”子程序中输入代码，同时新建“检查输入”子程序。代码如下：

子程序名	返回值类型	公开	备注
_添加按钮_被单击			

检查输入 ()

--- 如果 (添加按钮.标题 = “添加记录”)

添加按钮.标题 = “保存记录”

学号编辑框.获取焦点 ()

连续赋值 (“”, 学号编辑框.内容, 姓名编辑框.内容, 语文编辑框.内容, 数学编辑框.内容, 英语编辑框.内容)

如果 (外部数据库1.执行 (“INSERT INTO 成绩表 ” + “ (学号,姓名,语文,数学,英语)” + “ valueS ” + “ (” + 学号编辑框.内容 + “,” + 姓名编辑框.内容 + “,” + 语文编辑框.内容 + “,” + 数学编辑框.内容 + “,” + 英语编辑框.内容 + “)” , ) = 假)

--- 信息框 (“添加记录失败,请检查是否连接数据库,或填写类型是否适当”, 0, “添加失败”)

▶ 信息框 (“已成功添加为尾记录”, 0, “添加成功”)

外部数据库1.重新查询 (记录集句柄)

添加按钮.标题 = “添加记录”





子程序名	返回值类型	公开	备注
检查输入			

```

如果真 (学号编辑框.内容 = "")
    学号编辑框.内容 = "0"
如果真 (姓名编辑框.内容 = "")
    姓名编辑框.内容 = "?"
如果真 (语文编辑框.内容 = "")
    语文编辑框.内容 = "0"
如果真 (数学编辑框.内容 = "")
    数学编辑框.内容 = "0"
如果真 (英语编辑框.内容 = "")
    英语编辑框.内容 = "0"

```

为什么要加上一个“检查输入”子程序呢？因为下面所用的 SQL 指令要求所有项目都要输入，但有时有些字段不用输入，所以就用“？”或“0”号来代替。

(8) 双击“修改记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_修改按钮_被单击			

```

如果 (修改按钮.标题 = "修改记录")
    信息框 ("请直接修改编辑框的内容,然后按" + #左引号 + "保存修改" + #右引号, 0, "修改")
    姓名编辑框.获取焦点 ()
    修改按钮.标题 = "保存修改"
    如果 (外部数据库1.执行 ("UPDATE 成绩表 SET 姓名=" + "" + 姓名编辑框.内容 + "," + "语文=" + "" + 语文编辑框.内容 + "," + "数学=" + "" + 数学编辑框.内容 + "," + "英语=" + "" + 英语编辑框.内容 + " WHERE 学号=" + 学号编辑框.内容, ) = 假)
        信息框 ("修改失败!", 0, "失败")
        信息框 ("修改成功!", 0, "成功")
        外部数据库1.重新查询 (记录集句柄)
        修改按钮.标题 = "修改记录"

```

(9) 双击“删除记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_删除按钮_被单击			

```

如果真 (信息框 ("如果你真的想删除当前记录,请按" + #左引号 + "是" + #右引号, 4, "删除") = 5)
    如果 (外部数据库1.执行 ("DELETE * FROM 成绩表 " + "WHERE 学号=" + 学号编辑框.内容, ) = 假)
        信息框 ("删除失败!", 0, "失败")
        信息框 ("删除成功!", 0, "成功")
        外部数据库1.重新查询 (记录集句柄)
        删除按钮.标题 = "删除记录"
        显示记录 ()

```



(10) 查询记录。

插入一个窗口，添加一个“列表框”、“编辑框”、“组合框”和“按钮”组件。为组合框加入 5 个项目“学号”、“姓名”、“语文”、“数学”、“英语”。如图 12-26 所示。

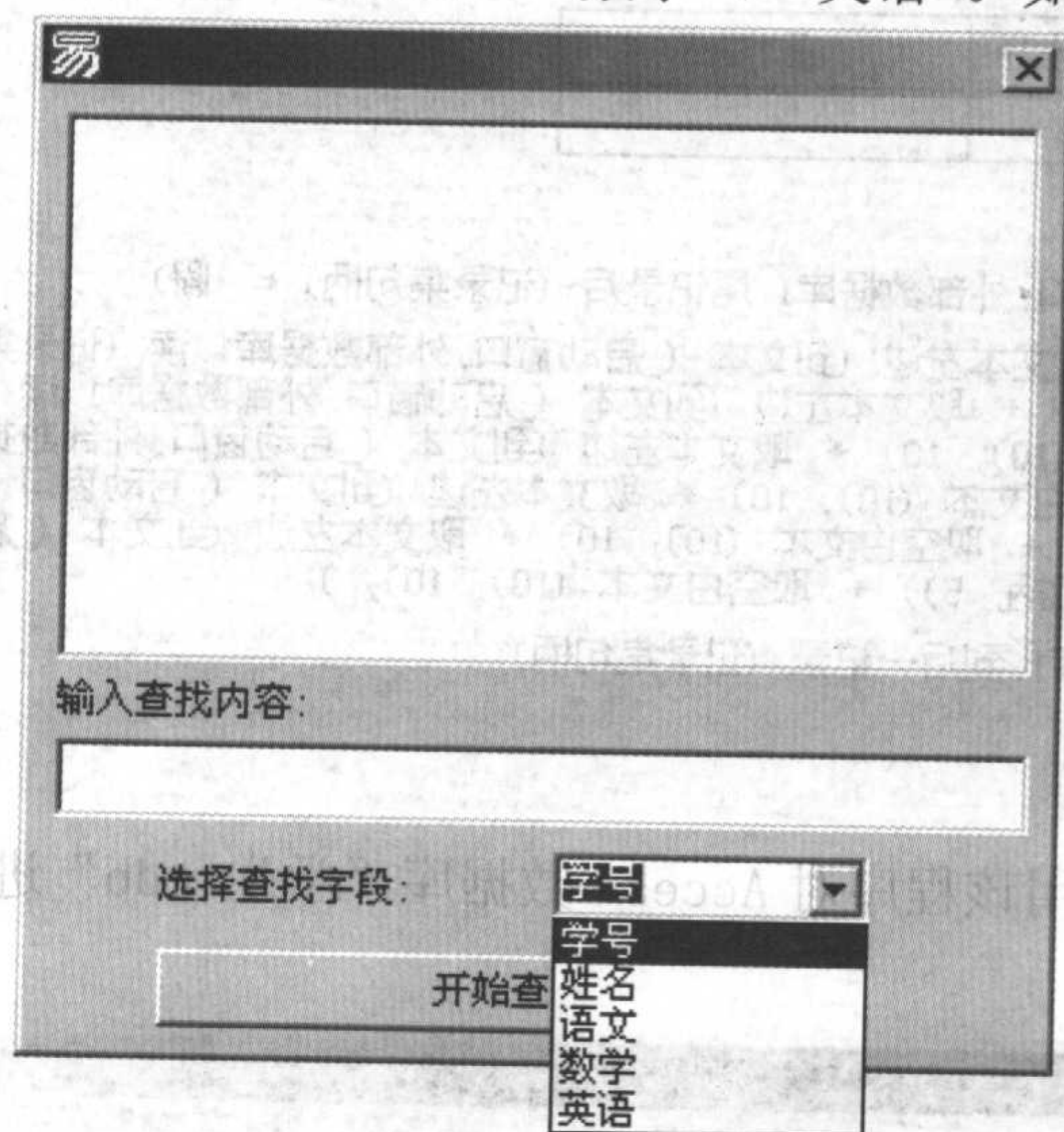


图 12-26 查询窗口界面

双击“\_启动窗口”中“查找记录”按钮，输入代码：

子程序名	返回值类型	公开	备注
_查找记录按钮_被单击			

载入 (查询窗口, , 真)

切换到“查询窗口”，双击“查询窗口”中的“开始查找”按钮，输入代码：

子程序名	返回值类型	公开	备注
_开始查找按钮_被单击			

\_启动窗口.外部数据库1.到首记录 (记录集句柄)

如果 (查找编辑框.内容 = "")

信息框 ("你还没有选择查找项目", 0, )

如果 (组合框1.现行选中项 = -1)

信息框 ("你还没有选择查找类型", 0, )

如果 (组合框1.内容 = "姓名")

记录集句柄 = \_启动窗口.外部数据库1.查询 ("select \* from 成绩表 where " + 组合框1.内容 + "=" + " " + 查找编辑框.内容 + " ")

记录集句柄 = \_启动窗口.外部数据库1.查询 ("select \* from 成绩表 where " + 组合框1.内容 + "=" + 查找编辑框.内容)

\_启动窗口.外部数据库1.重新查询 (记录集句柄)

显示记录 ()

\_启动窗口.外部数据库1.关闭记录集 (记录集句柄)





该程序集中，也建了一个“显示记录”子程序，用来将查找到的记录显示在列表框中，子程序代码如下：

子程序名	返回值类型	公开	备注
显示记录			

列表框1. 清空 ()

--> 判断循环首 (启动窗口.外部数据库1.尾记录后 (记录集句柄) = 假)

列表框1.加入项目 (取文本左边 (到文本 (启动窗口.外部数据库1.读 (记录集句柄, 1)) + 取空白文本 (7), 7) + 取文本左边 (到文本 (启动窗口.外部数据库1.读 (记录集句柄, 2)) + 取空白文本 (10), 10) + 取文本左边 (到文本 (启动窗口.外部数据库1.读 (记录集句柄, 3)) + 取空白文本 (10), 10) + 取文本左边 (到文本 (启动窗口.外部数据库1.读 (记录集句柄, 4)) + 取空白文本 (10), 10) + 取文本左边 (到文本 (启动窗口.外部数据库1.读 (记录集句柄, 5)) + 取空白文本 (10), 10), )

\_启动窗口.外部数据库1.到后一记录 (记录集句柄)

-- 判断循环尾 ()

3. 试运行程序，使用该程序对 Access 数据库“学生.mdb”进行操作，如图 12-27 所示。

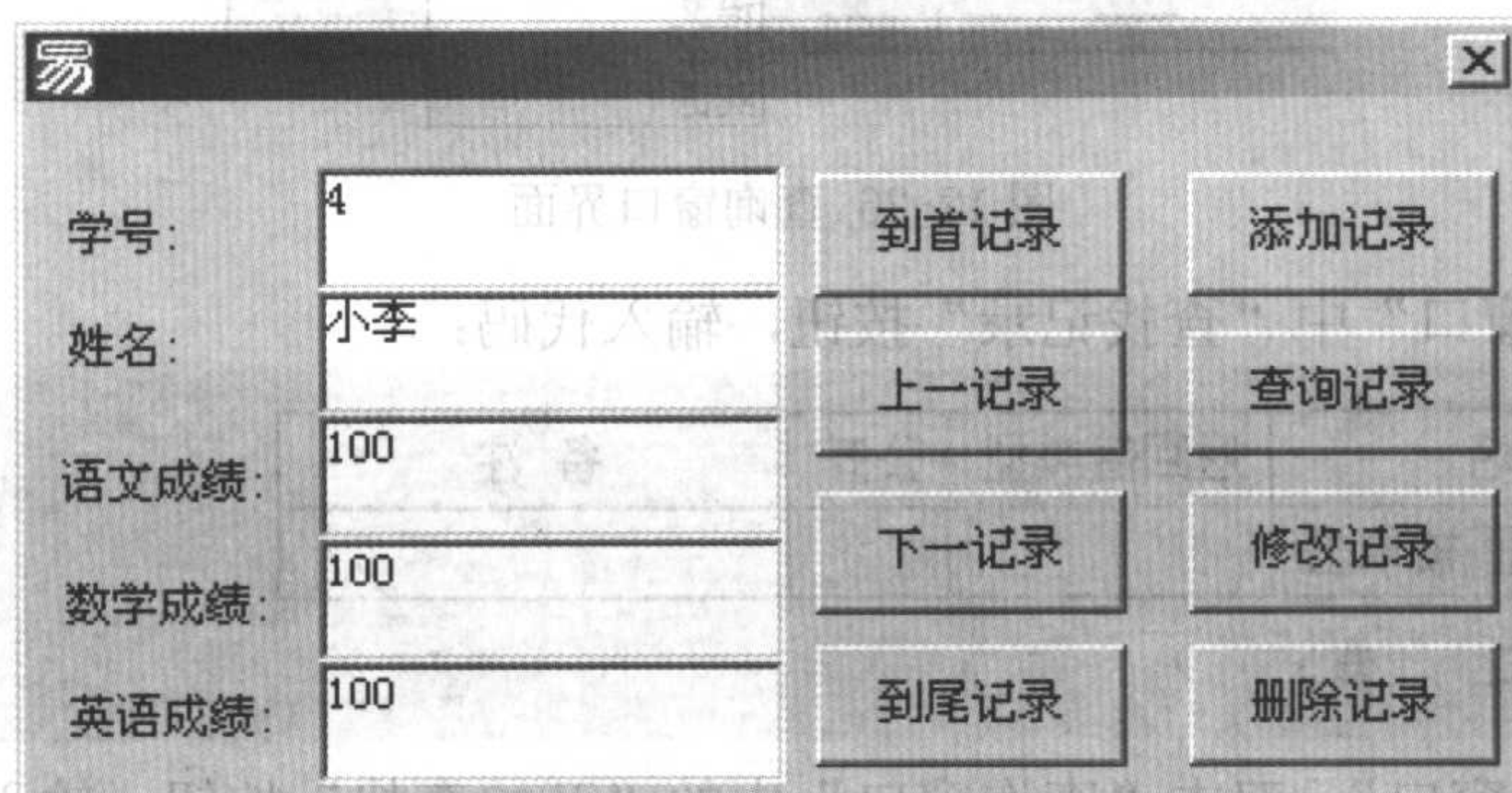


图 12-27 对 Access 数据库进行操作

例程参见随书光盘中“access 数据库操作.e”。

## 12.6 SQL Server 数据库

### 12.6.1 SQL Server 简介

Microsoft SQL Server 由 Microsoft 公司开发，搭建了一个功能强大的客户/服务器平台，能够为各种环境的数据库应用软件提供服务。

SQL Server 2000 由两个部分组成：服务器组件和客户端工具。

#### 1. 服务器组件

SQL Server 的服务器组件是以 Windows 服务 (Windows Services) 方式运行的。一般认为 SQL Server 包含四种 Windows 服务 (这里大家关注 OLTP、暂时不考虑 OLAP)，分别



是: MS Sql Server、DTC Distributed Transaction coordinator、SQLServerAgent、Search Service。

MSSqlServer 是最常用的服务,一般的数据库功能都是由它提供的,例如文件管理、查询处理、数据存储等;DTC 是分布式事务协调器,支持跨越两个或多个服务器的更新操作来保证事务的完整性;SQLServerAgent 负责 SQL Server 自动化工作,如果需要 SQL Server 在指定时间执行某一个存储过程,就需要用到这个服务了;Search Service 是全文查询服务,负责全文检索方面的工作。

## 2. 客户端工具

SQL Server 2000 的核心是上文讨论的那些服务器组件,但用户直接接触的却不是它们,而是客户端工具。服务器组件是引擎,客户端工具是用户界面,两者是相辅相成的。

先来看一下 SQL Server 2000 的客户端工具到底有哪些——企业管理器、查询分析器、事件探查器、服务管理器、客户端网络实用工具、服务器网络实用工具、导入和导出数据(DTS)等等。

服务器组件与客户端工具功能上是配套的,客户端工具需要用最简单的形式表达最丰富的服务器组件的功能;服务器组件和客户端工具物理上是离散的,说句大白话:它们不是同一个程序!客户端工具要与服务器组件连通,需要一些用于通讯的动态链接库,SQL Server 2000 的通讯库支持多种网络协议,例如 TCP/IP、命名管道等。

只要客户端工具与服务器组件与功能上是兼容的,就可以通过一定的协议连接,所以只要在自己的机器上装一套客户端工具,就可以连接世界各地的 SQL Server 服务器,当然这需要对方开放足够的权限。

**提示:** 安装 SQL Server 2000 实际上就是安装服务器组件和客户端工具。当然,大家可以选择同时安装服务器组件和客户端工具,或者只安装其中的一个,甚至只选择安装更少的东西。

## 3. 版本问题

SQL Server 2000 有很多版本:企业版、开发版、标准版、个人版等。每一个版本包含的客户端工具基本上是一样的,而服务器组件可能有些不同。所以在安装 SQL Server 之前必须参考表 12-7 操作系统和 SQL Server 版本兼容表,根据操作系统选择合适的 SQL Server 版本。

操作系统 版本	Windows 2000 NT 4(sp5)	Windows 2000 Professional、NT4 workstation(sp5)、 windows XP	Windows 98、 windows Me
企业版			
开发版			
标准版			
个人版			

表 12-7 操作系统和 SQL Server 版本兼容表





## 12.6.2 SQL Server 2000 安装

SQL Server 2000 的安装程序是非常智能化的，基本上用户只需要跟着提示，选择默认项。下面以标准版为例挑选一些需要注意的设定画面进行讲解。

首先将 SQL Server 2000 的光盘放入光驱，会自动出现安装界面，如图 12-28 所示。

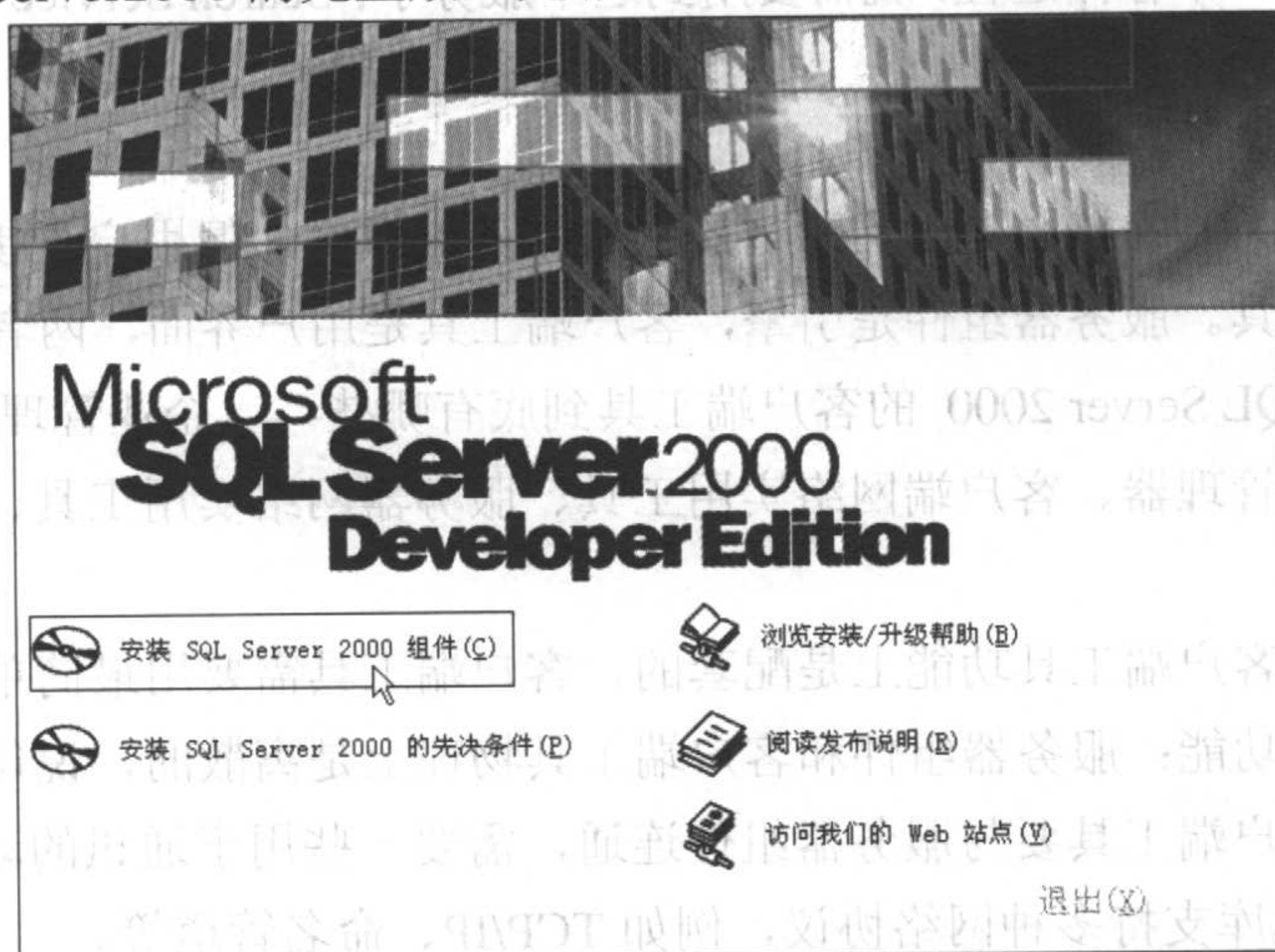


图 12-28 SQL Server 安装界面

选择“安装 SQL Server 组件”，在接下来弹出的界面中选择“安装数据库服务器”。选择后自动进入 SQL Server 的安装程序。安装程序中，开始的对话框都使用默认选项，并点击“下一步”按钮。当出现“软件许可证协议”窗口，单击“是”按钮。在接下来弹出的“安装定义”窗口，选择“服务器和客户端工具”选项，然后点击“下一步”按钮，如图 12-29 所示。

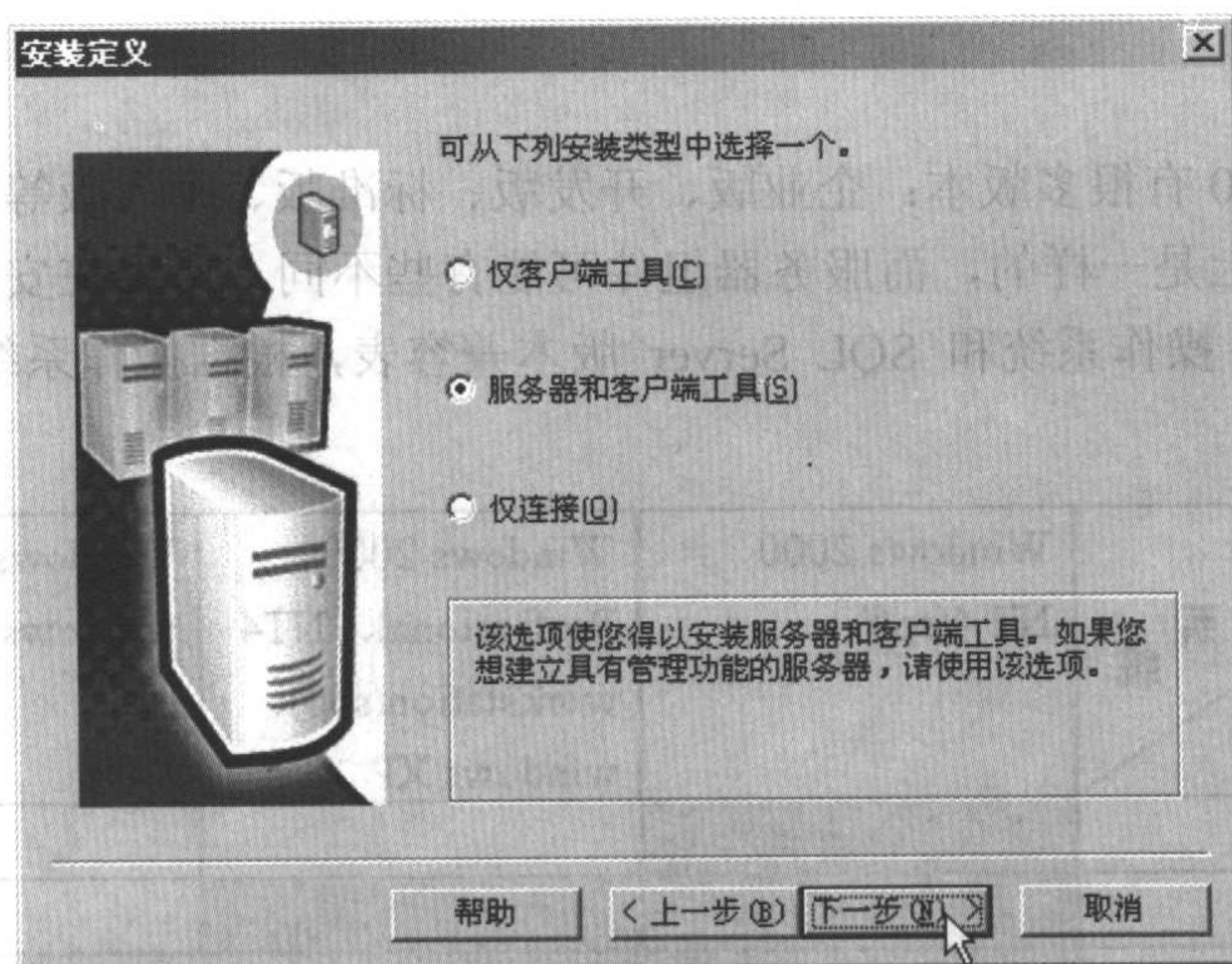


图 12-29 定义欲安装的组件

接下来会弹出“实例名”窗口，如果是首次安装，直接选择默认实例名，否则系统会提示您输入，之后点击“下一步”。然后会弹出“安装类型”窗口，选择“典型”安装类



型即可，默认安装位置，如图 12-30 所示。

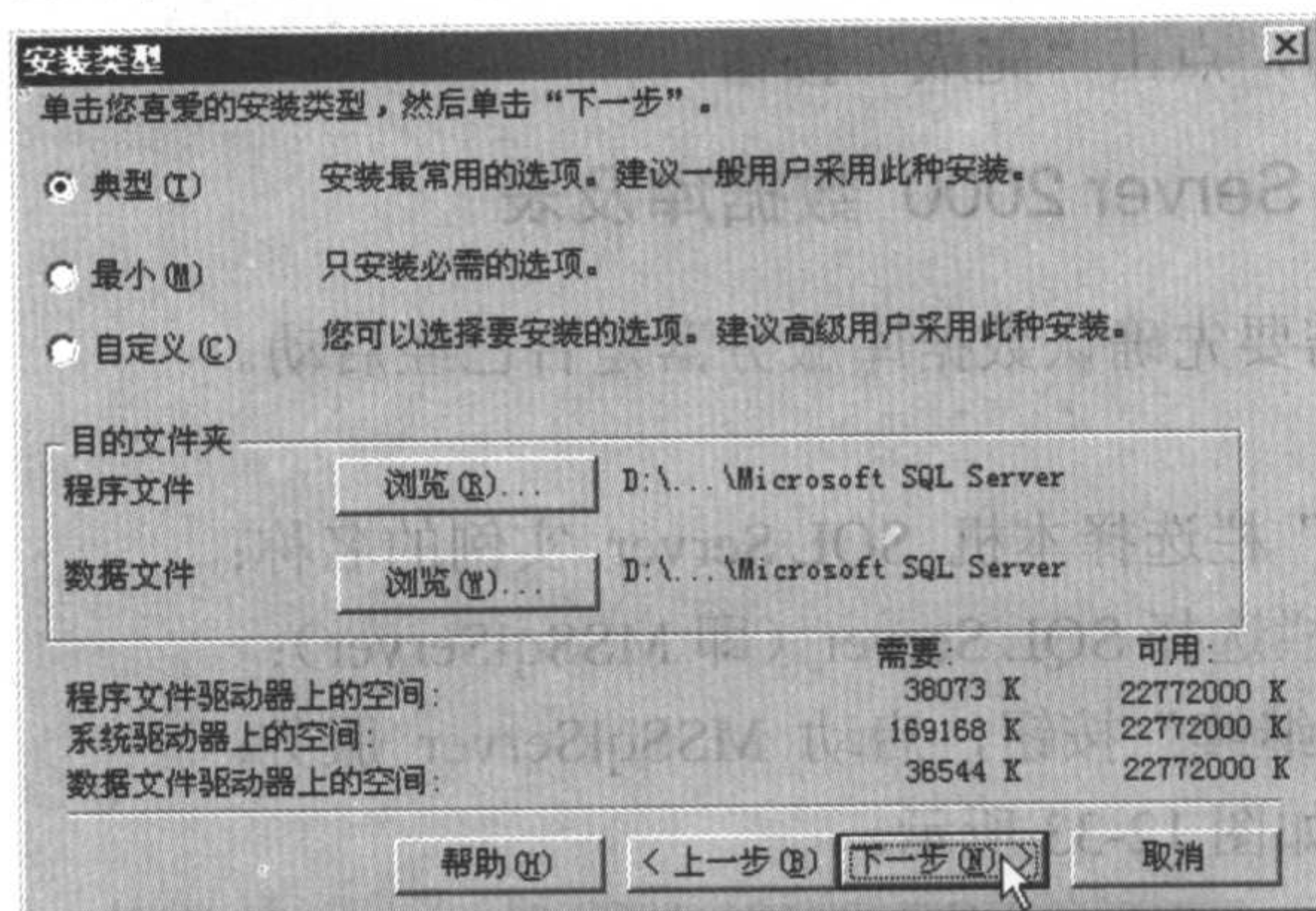


图 12-30 安装配置

下面会弹出“服务账户”窗口，可以为服务设定启动方式，这里有两个服务，分别是 SQL Server 和 SQL Server 代理。这里大家操作系统以“本地系统账户”启动它们，如图 12-31 所示。

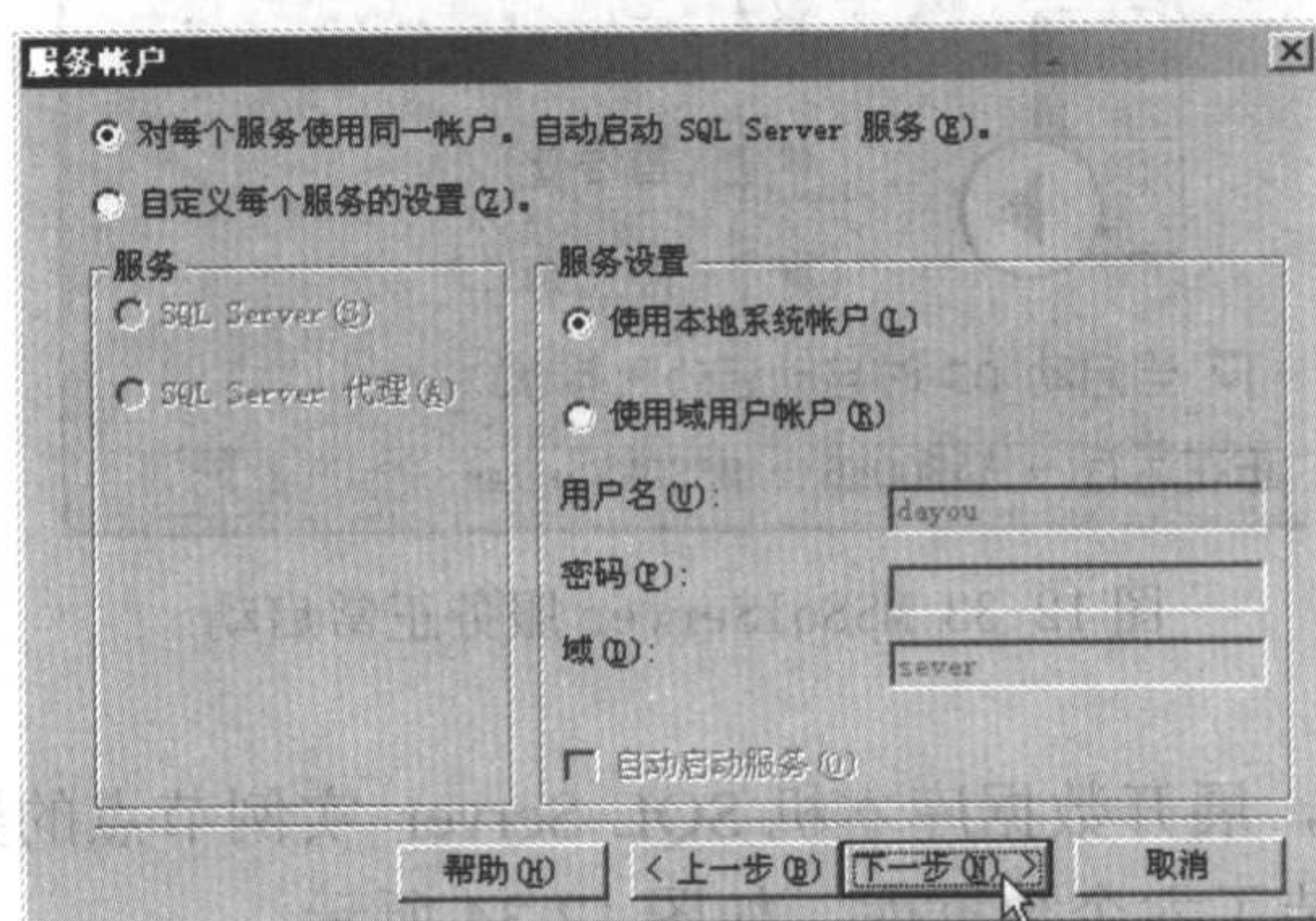


图 12-31 选择服务账户

弹出“身份验证模式”窗口，初学可以直接选择混合模式验证，设置空密码即可如图 12-32 所示。

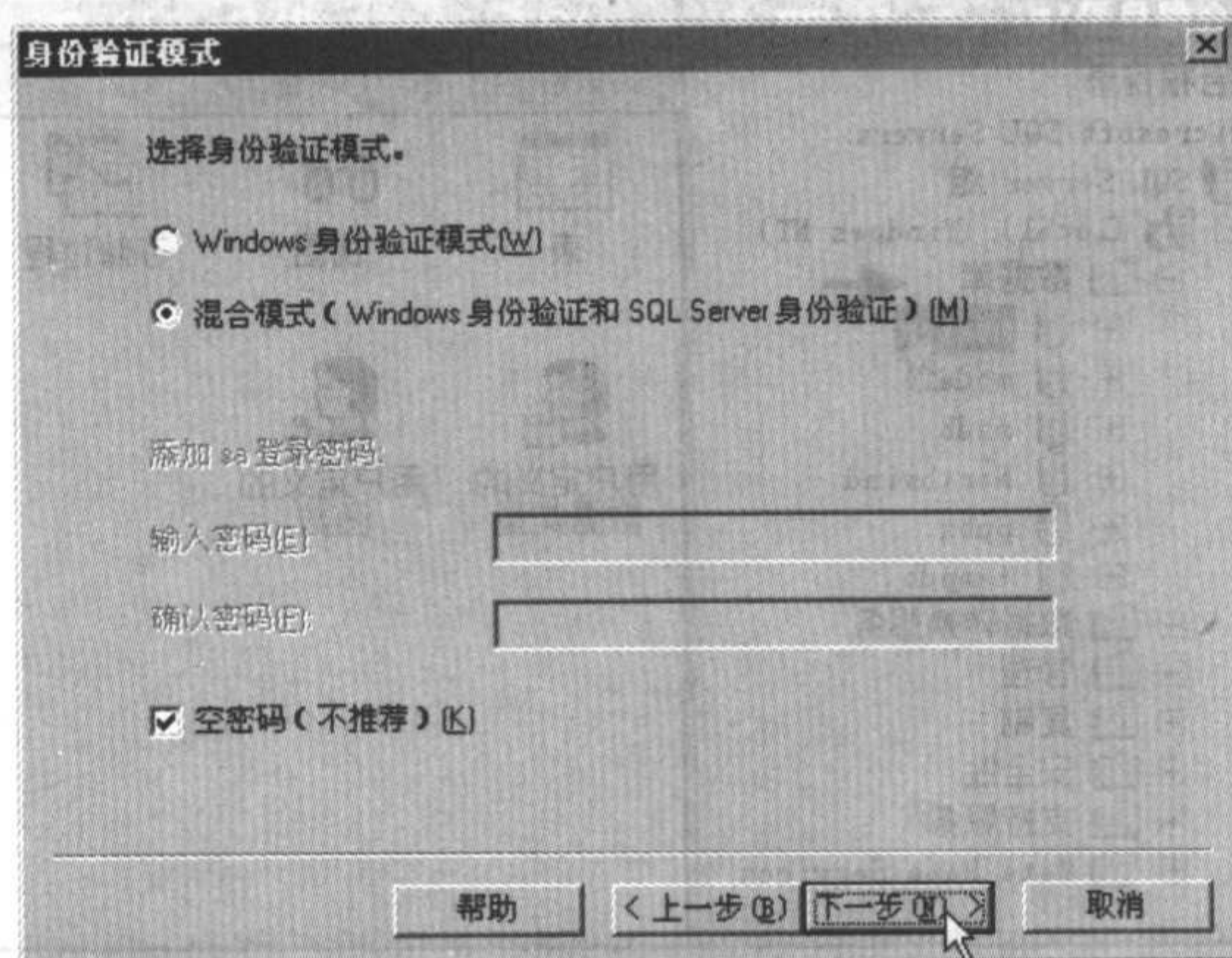


图 12-32 身份验证模式





接下来的窗口都使用默认配置，直接点击“下一步”按钮和“继续按钮”。然后 SQL Sever 开始安装，安装完毕后，点击“完成”按钮。

## 12.6.3 创建 SQL Server 2000 数据库及表

安装完毕，大家需要先确认数据库服务器是否已经启动。

### 1. 服务管理器

- (1) 在“服务器”栏选择本机 SQL Server 实例的名称；
- (2) 在“服务”栏选择 SQL Server (即 MSSqlServer)；
- (3) 按下“开始/继续”按钮，启动 MSSqlServer 服务；
- (4) 正常启动，如图 12-33 所示。



图 12-33 MSSqlServer 服务正常启动

### 2. 企业管理器

打开“企业管理器”，展开数据库本机 SQL Server 实例节点的数据库项目，可以发现 SQL Server 已经自动安装了六个数据库。如图 12-34 所示。

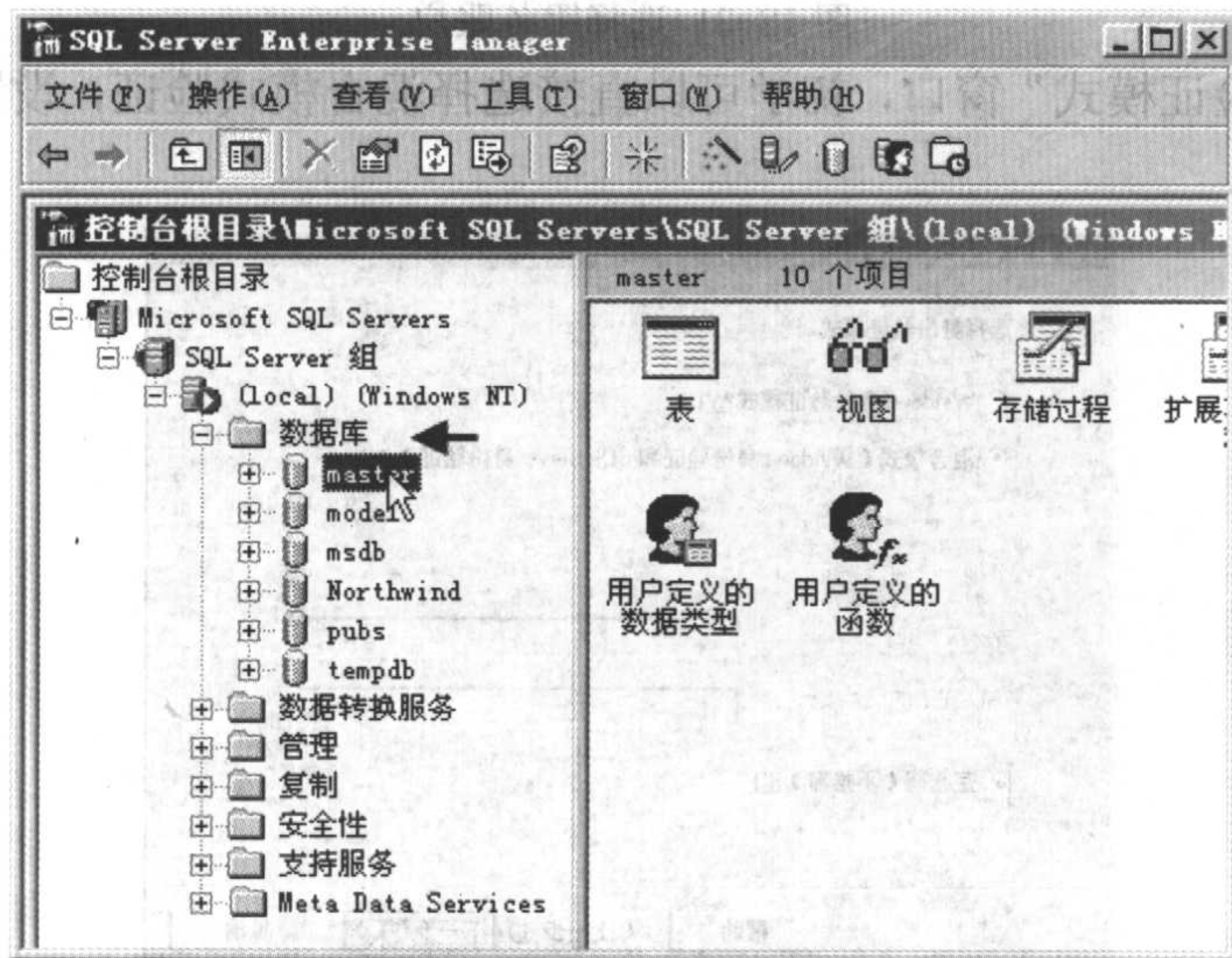


图 12-34 自动安装的数据库



- master 数据库记录 SQL Server 系统的所有系统级别信息。
- tempdb 数据库保存所有的临时表和临时存储过程。
- model 数据库是为用户创建数据库提供的模板。
- msdb 数据库供 SQL Server 代理程序调度警报和作业以及记录各种操作。
- northwind 和 pubs 是示例数据库，以后大家要经常利用它们做演示。

除了 northwind 和 pubs 之外，其他四个数据库都是由 SQL Server 自行维护的，一般不需要用户管理。

### 3. 创建数据库及表

在 SQL Server 中，可以使用企业管理器来创建数据库和表。

创建 SQL Server 数据库可以将鼠标放在企业管理器中的数据库项上，然后点击鼠标右键，在弹出菜单中选择“新建数据库”。如图 12-35 所示。



图 12-35 创建新数据库

点击“新建数据库”后，会弹出“数据库属性”对话框，在对话框中输入要创建数据库的名称。如图 12-36 所示。

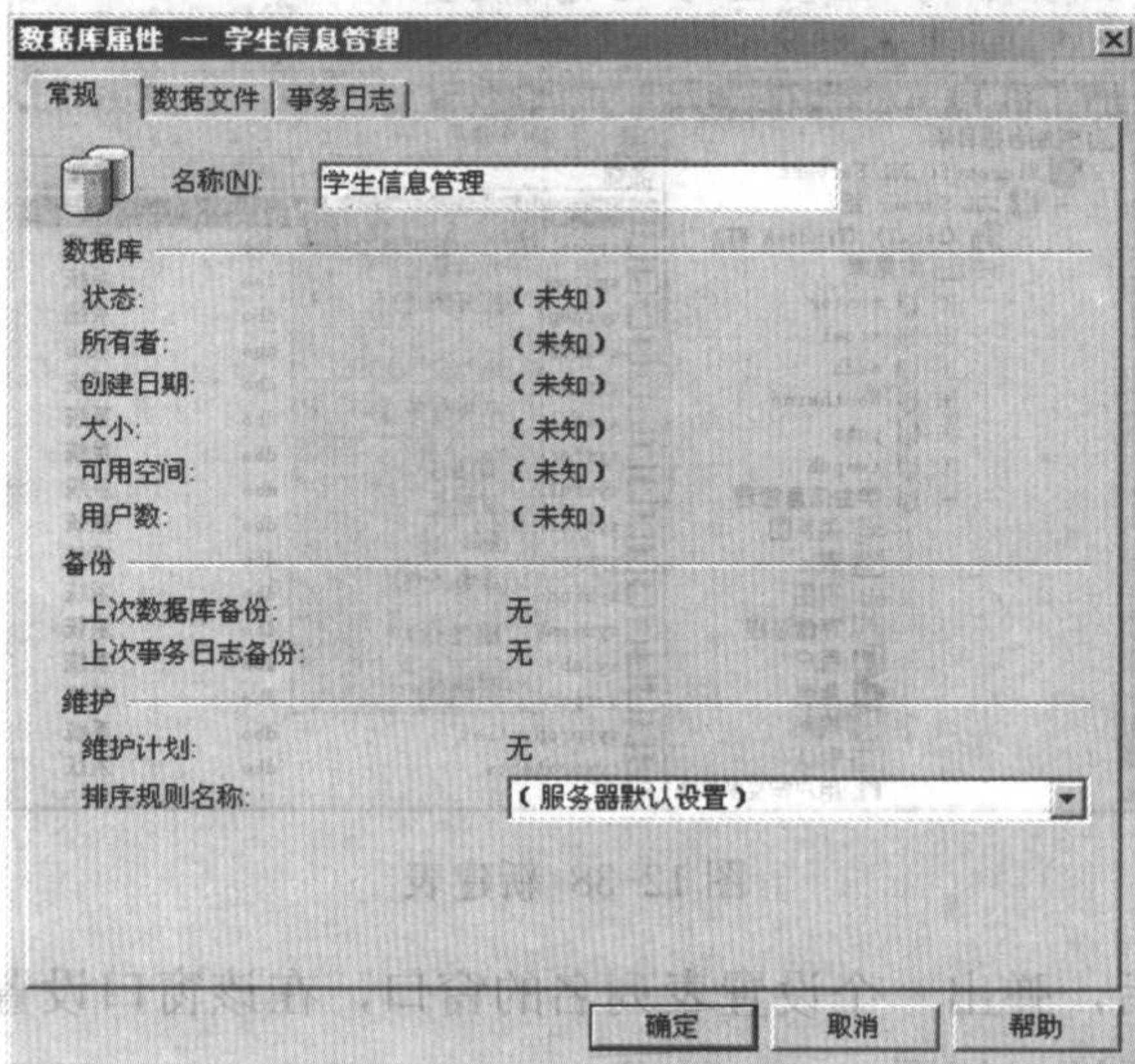


图 12-36 设置数据库属性





这里，创建一个“学生信息管理”数据库，后面将用易语言对这个数据库进行操作。点击“数据库属性”窗口中的“数据文件”选项卡，可以在“数据库”文件子夹中设置数据库保存的目录和文件名。如图 12-37 所示。

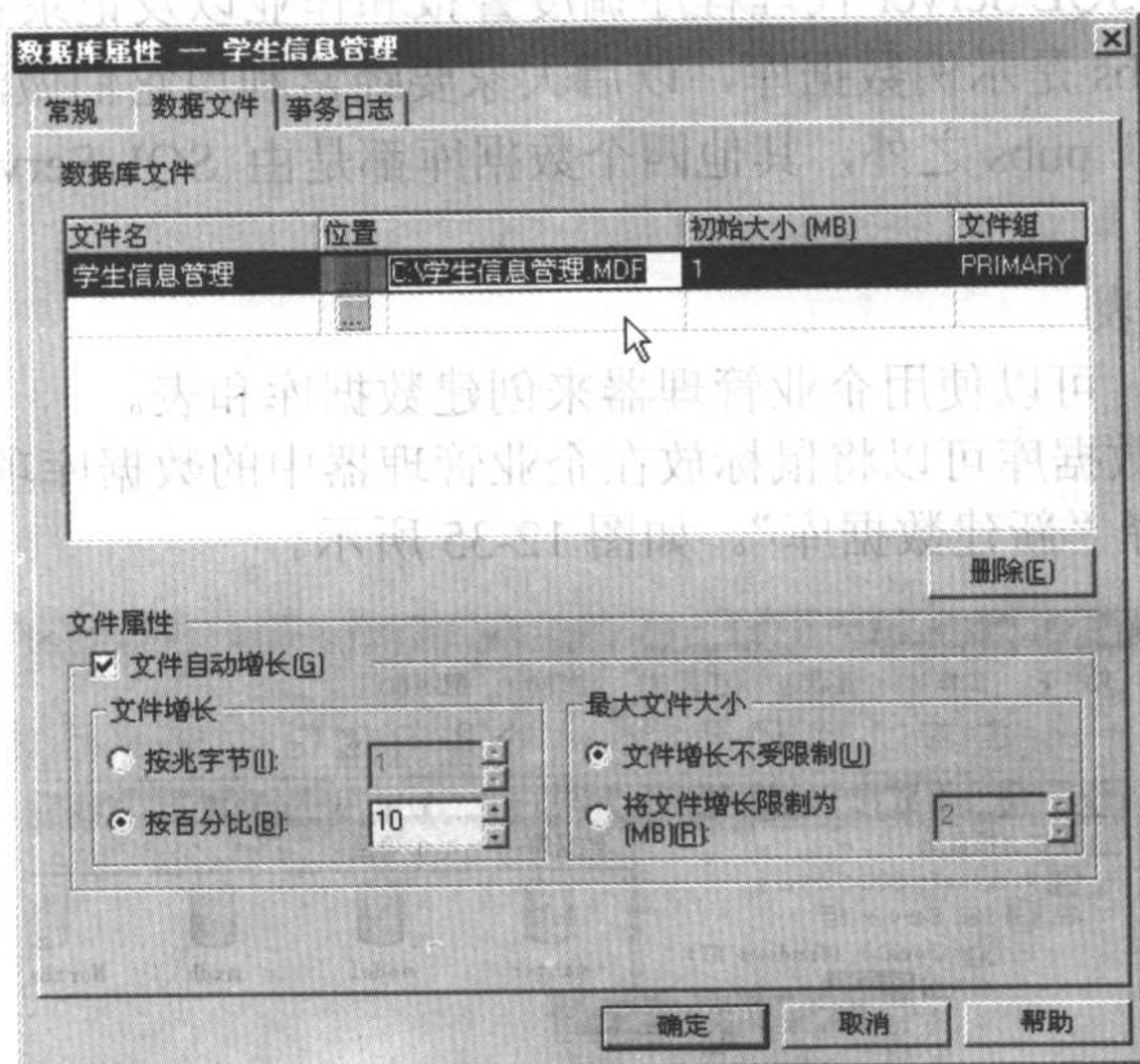


图 12-37 设置数据库保存的文件名和目录

这里，将数据库文件名设置为“学生信息管理.MDF”，将保存目录设置为 C 盘。最后点击确定按钮，就可以创建一个新的数据库了。

下面就要在新建的数据库中创建一个新表，首先展开控制台中新建的“学生信息管理”数据库，选中“表”，然后在右面的列表中点击鼠标右键，选择弹出菜单中的“新建表”。如图 12-38 所示。

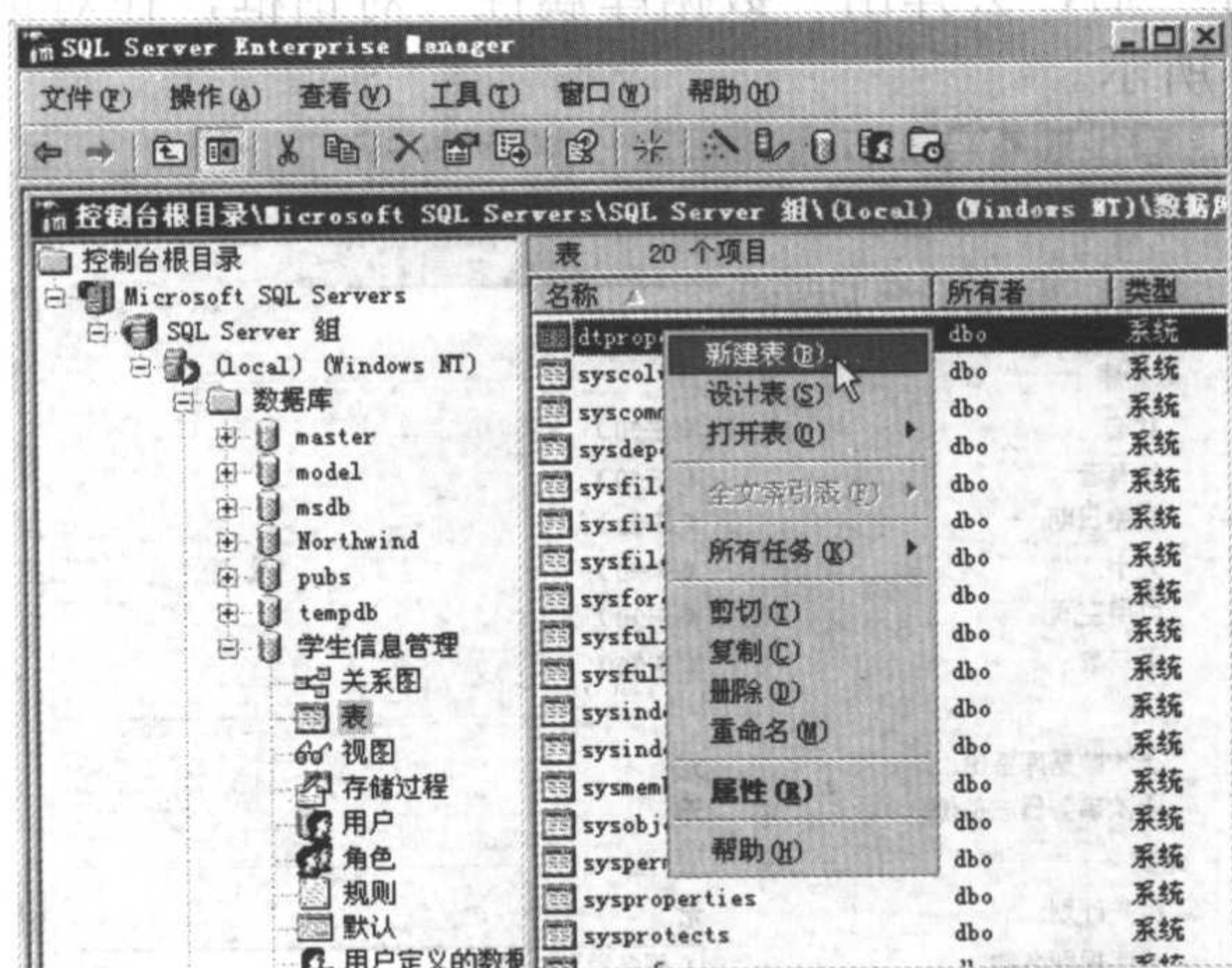


图 12-38 新建表

选择“新建表”后，弹出一个设置表列名的窗口，在该窗口设置新建表的字段名、字段的数据类型及其他属性。如图 12-39 所示。



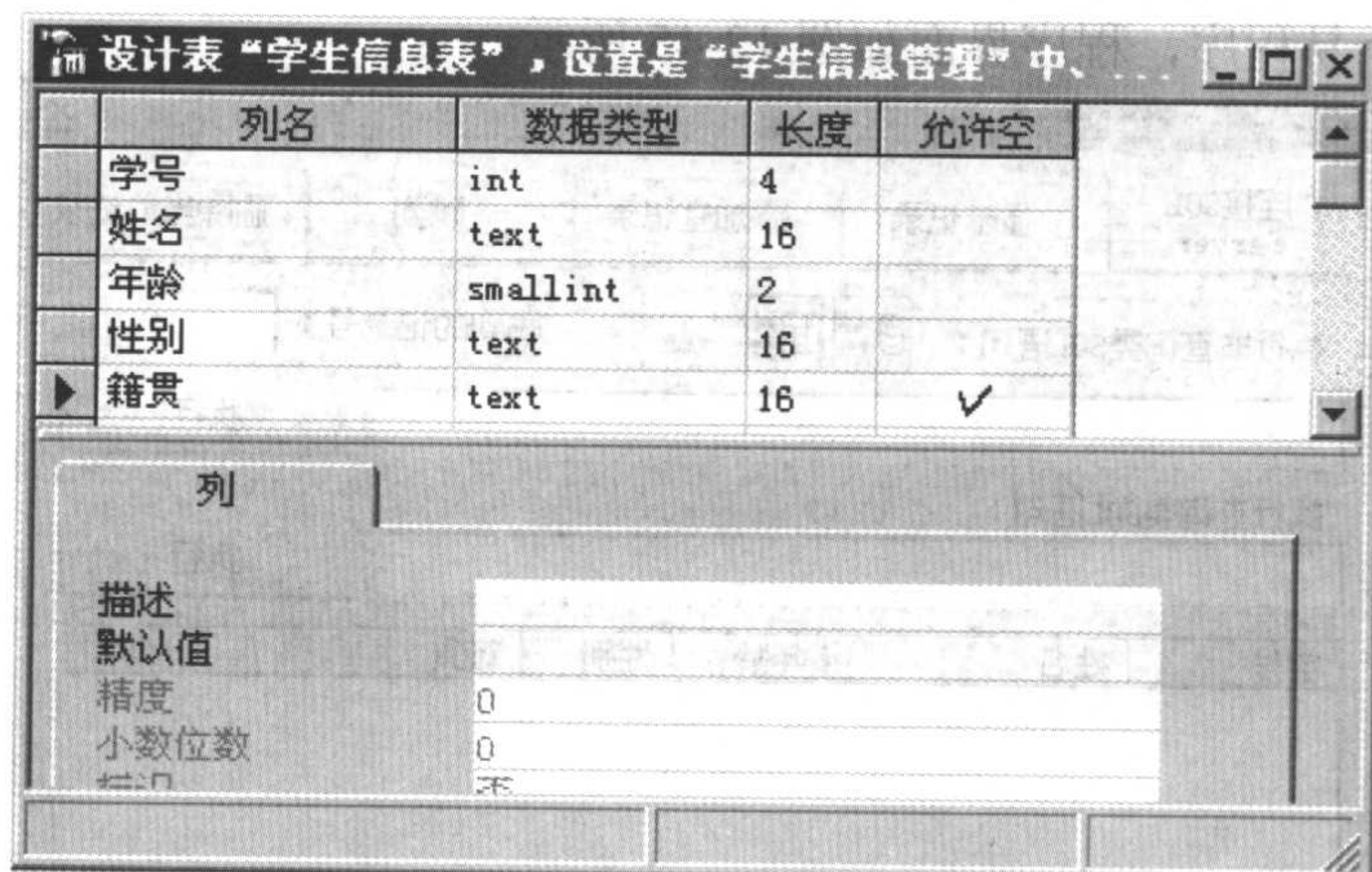


图 12-39 创建表列

创建字段后关闭窗口，会弹出“是否保存”信息框，点击“是”，然后在接下来弹出的输入框中输入欲保存的表名，本例保存为“学生信息表”。

创建后会在列表中找到新建的表，在新建表上点击鼠标右键，然后选择弹出菜单中“打开表”→“返回所有行”，会弹出添加记录窗口，在窗口中加入一些记录，如图 12-40 所示。

学号	姓名	年龄	性别	籍贯
2003001	钱开	22	男	山东
2003002	张强	20	男	山东
2003003	王一	20	男	湖南
2003004	李杰	21	女	湖北
2003005	朱明	23	男	上海
2003006	赵直	20	男	北京
2003007	孙新	21	女	广东
2003008	李明	22	男	内蒙
2003009	周柯	20	女	新疆
2003010	张新	22	女	澳门

图 12-40 向表中加入记录

加入记录后关闭窗口，SQL Server 会自动保存记录。

#### 12.6.4 使用易语言操作 SQL SERVER 数据库

可以使用易语言中的外部数据库组件对 SQL Server 数据库进行操作，前面使用“外部数据库”和“外部数据库提供者”对 Access 数据库进行过操作，使用相同方法也可以对 SQL Server 数据库进行操作，这里介绍一下用“数据库连接”和“记录集”组件对 SQL Server 数据库进行操作。

下面，就使用易语言对“学生信息管理”数据库进行操作。





首先新建一个易程序，程序界面如图 12-45 所示。

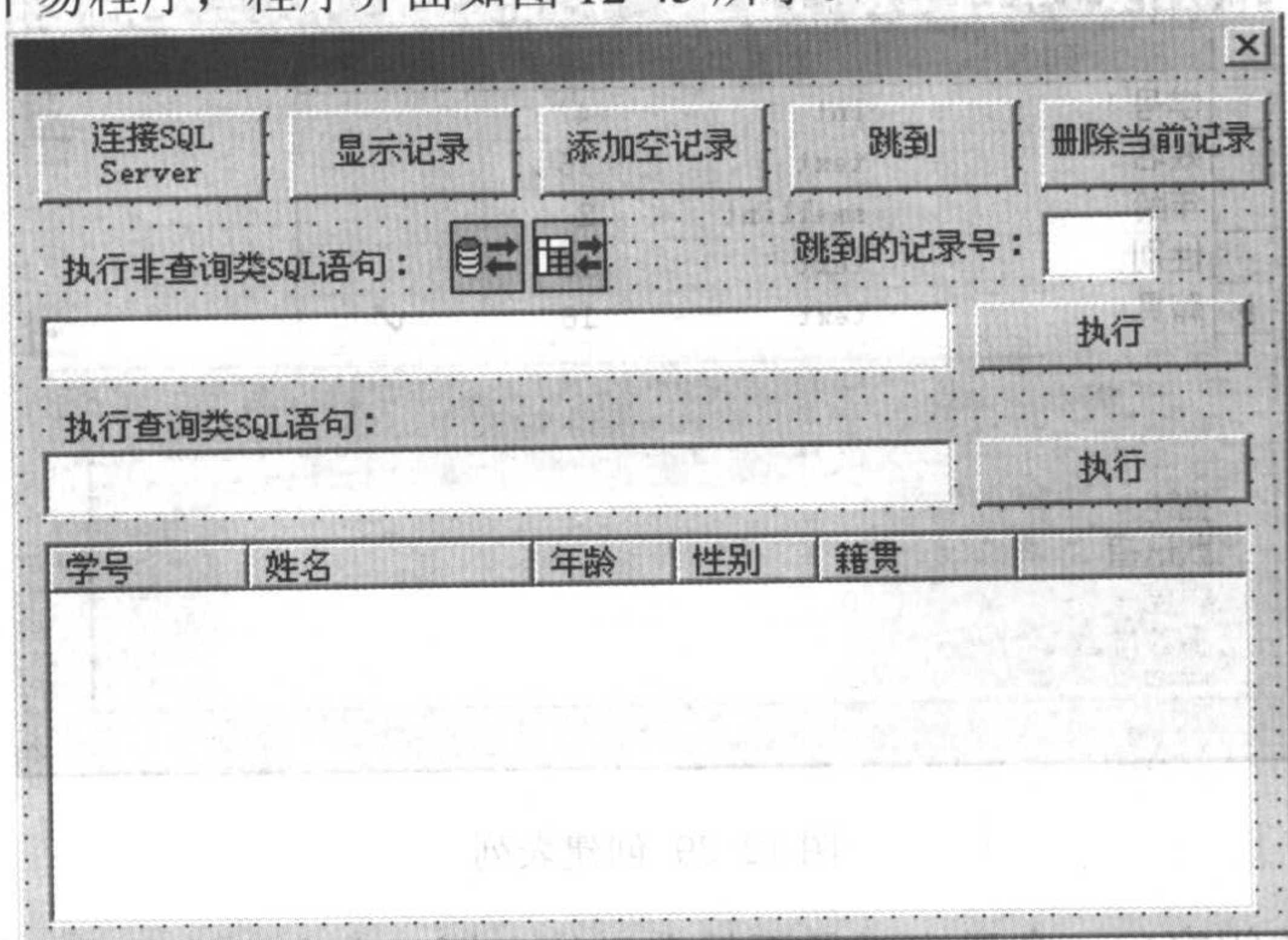


图 12-45 程序界面设计

窗口中，添加了“数据库连接”和“记录集”组件，还添加了 1 个“超级列表框”组件，将“超级列表框”组件的“类型”属性设置为“报表列表框”，并在“报表列”属性中加入了如图的 5 个报表列，其他的组件是大家熟悉的“标签”、“编辑框”和“按钮”组件，该例程中都将按钮的“名称”属性改为和其功能对应的名称。

在对 SQL Server 数据库进行操作前，首先要成功的连接 SQL Server 数据库，所以，第一步，双击“连接 SQL Server”按钮，在“\_连接按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_连接按钮_被单击			

```
如果 (数据库连接1.连接SQLServer ("192.168.0.66", "学生信息管理",  
    "sa", "") = 真)  
    如果 (记录集1.置连接 (数据库连接1) = 真)  
        信息框 ("连接成功", 0, )  
        信息框 ("记录集置连接失败", 0, )  
    信息框 ("连接失败", 0, )
```

程序中，使用了 SQL Server 数据库的缺省用户名“sa”，密码为空。用“数据库连接”组件的“连接 SQL Server”的方法，连接 SQL Server 数据库，在成功连接后，使用“记录集”组件“置连接”方法，将“记录集”组件和“数据库连接”组件相连，在方法的参数中直接填写“数据库连接”组件的名称。

**注意：**连接 SQL Server 数据库也可以使用“数据库连接”组件的“连接”方法。

第二步，新建一个“加入列”子程序，用来动态的给超级列表框加列，由于程序会执行查询类 SQL 语句，查询的记录不一定是全部字段，所以，每次用超级列表框显示记录前



先给超级列表框的列全部删除，然后重新加入列，记录集中有几个字段，就为超级列表框加入几列。代码如下：

子程序名	返回值类型	公开	备注
加入列			

变量名	类型	静态	数组	备注
循环变量	整数型			

```

超级列表框1.全部删除 ()
--▶ 计次循环首 (超级列表框1.取列数 (), 循环变量)
    超级列表框1.删除列 ()
-- 计次循环尾 ()
--▶ 计次循环首 (记录集1.字段数量, 循环变量)
    超级列表框1.插入列 (-1, 记录集1.取字段名 (循环变量 - 1), , , )
-- 计次循环尾 ()
    
```

上述程序中，使用了2个循环，第一个循环用来删除所有列，另一个循环用来加入列，加入列的时候用“记录集”组件的“取字段名”方法，用返回的字段名来命名超级列表框的列标题。

第三步，新建一个“显示记录”子程序，用来将打开的记录集显示在超级列表框中，子程序代码如下：

子程序名	返回值类型	公开	备注
显示结果			

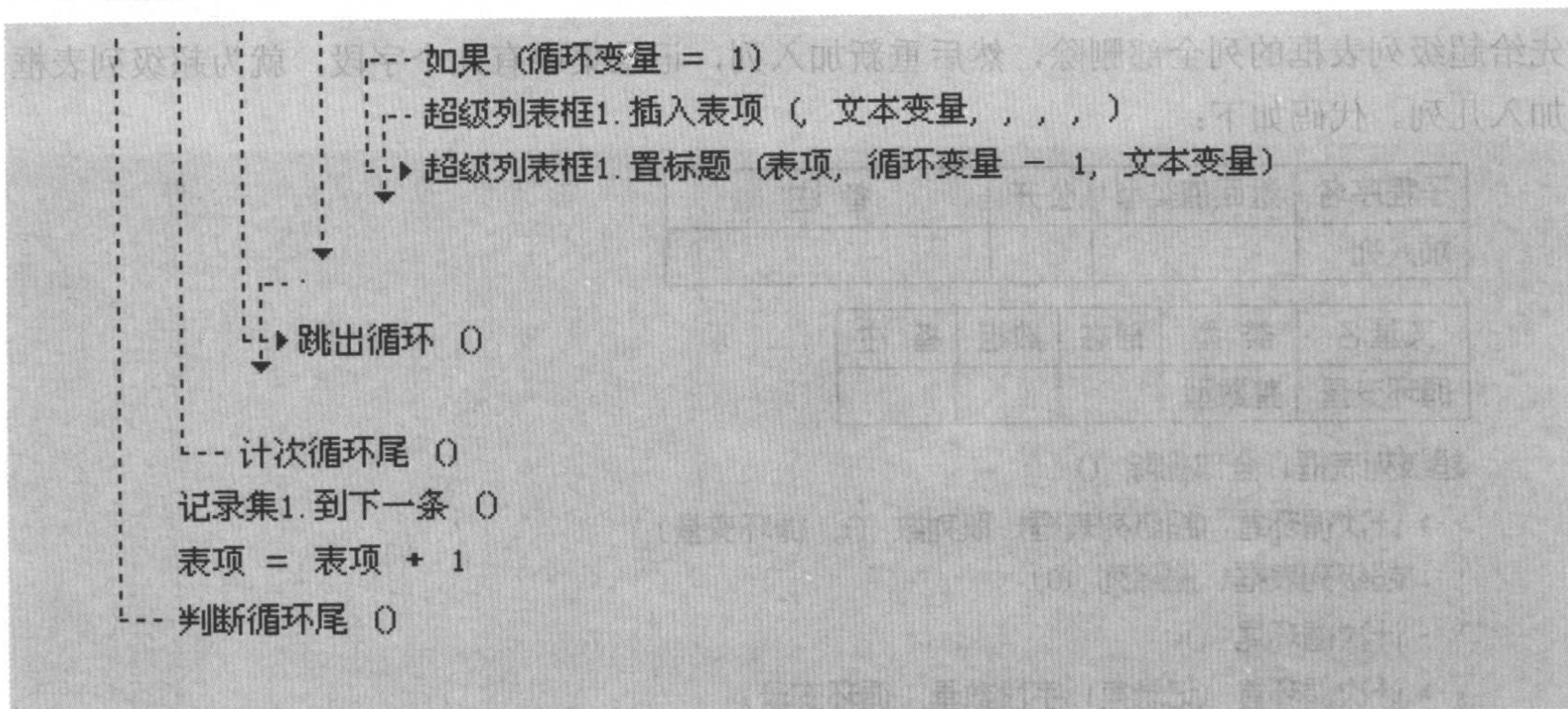
  

变量名	类型	静态	数组	备注
循环变量	整数型			
整数变量	整数型			
文本变量	文本型			
表项	整数型			

```

记录集1.到首记录 ()
--▶ 判断循环首 (记录集1.尾记录后 = 假)
    --▶ 计次循环首 (记录集1.字段数量, 循环变量)
        -- 如果 (记录集1.取字段名 (循环变量 - 1) ≠ "")
            -- 如果 (记录集1.取字段类型 (循环变量 - 1) = #整数型字段)
                记录集1.读整数 (循环变量 - 1, 整数变量)
                -- 如果 (循环变量 = 1)
                    超级列表框1.插入表项 ( 到文本 (整数变量), , , )
                    超级列表框1.置标题 (表项, 循环变量 - 1, 到文本 (整数变量))
                -- 记录集1.读文本 (循环变量 - 1, 文本变量)
            -- 记录集1.读文本 (循环变量 - 1, 文本变量)
        -- 计次循环尾 ()
    -- 判断循环尾 ()
    
```





由于“记录集”返回的字段数量不固定，所以程序中使用了“计次循环首（）”命令，来显示当前记录的所有字段内容，使用了“记录集”的“字段数量”属性，根据字段数量规定“计次循环首（）”命令的循环次数。

“记录集”读记录的时候，要注意字段的数据类型，根据字段类型选择“记录集”的方法，例如文本型字段用“读文本（）”方法，整数型字段用“读整数（）”方法。

第三步，双击“显示记录”按钮，在“\_显示记录按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_显示记录按钮_被单击			

记录集1.打开（“学生信息表”，#数据表名）

加入列（）

显示结果（）

程序中用“记录集”的“打开（）”方法，将“学生信息表”打开，并调用“显示记录”子程序，将打开的表显示在超级列表框中。

第四步，双击“添加空记录”按钮，在“\_添加空记录按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_添加空记录按钮_被单击			

记录集1.打开（“学生信息表”，#数据表名）

信息框（选择（记录集1.添加（），“添加成功”，“添加失败”），0，）

第五步，双击“跳到”按钮，在“\_跳到按钮\_被单击”子程序中输入代码：



子程序名	返回值类型	公开	备注
_跳到按钮_被单击			

记录集1. 打开 (“学生信息表”, #数据表名)

记录集1. 移到 (到数值 (编辑框3. 内容))

程序中使用了“记录集”的“移到 ( )”方法，可以将记录集的当前记录指针移动到指定记录处。

第六步，双击“删除记录”按钮，在“\_删除记录按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_删除记录按钮_被单击			

记录集1. 删除 (1)

“删除 ( )”方法的参数可以选填 1 或 3，若选择 1 则是删除当前记录，选择 3 则是删除全部记录。

第七步，双击第一个“执行”按钮，在“\_执行按钮 1\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_执行按钮1_被单击			

--- 如果 (数据库连接1. 执行SQL (编辑框1. 内容) = 真)

--- 信息框 (“执行成功”, 0, )

--- 信息框 (“执行失败”, 0, )

程序中，用“数据库连接”的“执行 SQL ( )”方法，可以执行 SQL 语句，该方法只返回是否执行成功。

第八步，双击第二个“执行”按钮，在“\_执行按钮 2\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_执行按钮2_被单击			

记录集1. 打开 (编辑框2. 内容, #SQL语句)

加入列 ()

显示结果 ()

“记录集”将打开查询类 SQL 语句返回的记录集，并用“显示记录”子程序将查询结果显示在“超级列表框”中。

最后试运行程序，首先连接 SQL Server 数据库，连接成功后，可以使用按钮查看“学生信息表”的内容，大家还可以用前面讲到的 SQL 语句对数据库进行各种操作。如图 12-46 所示。



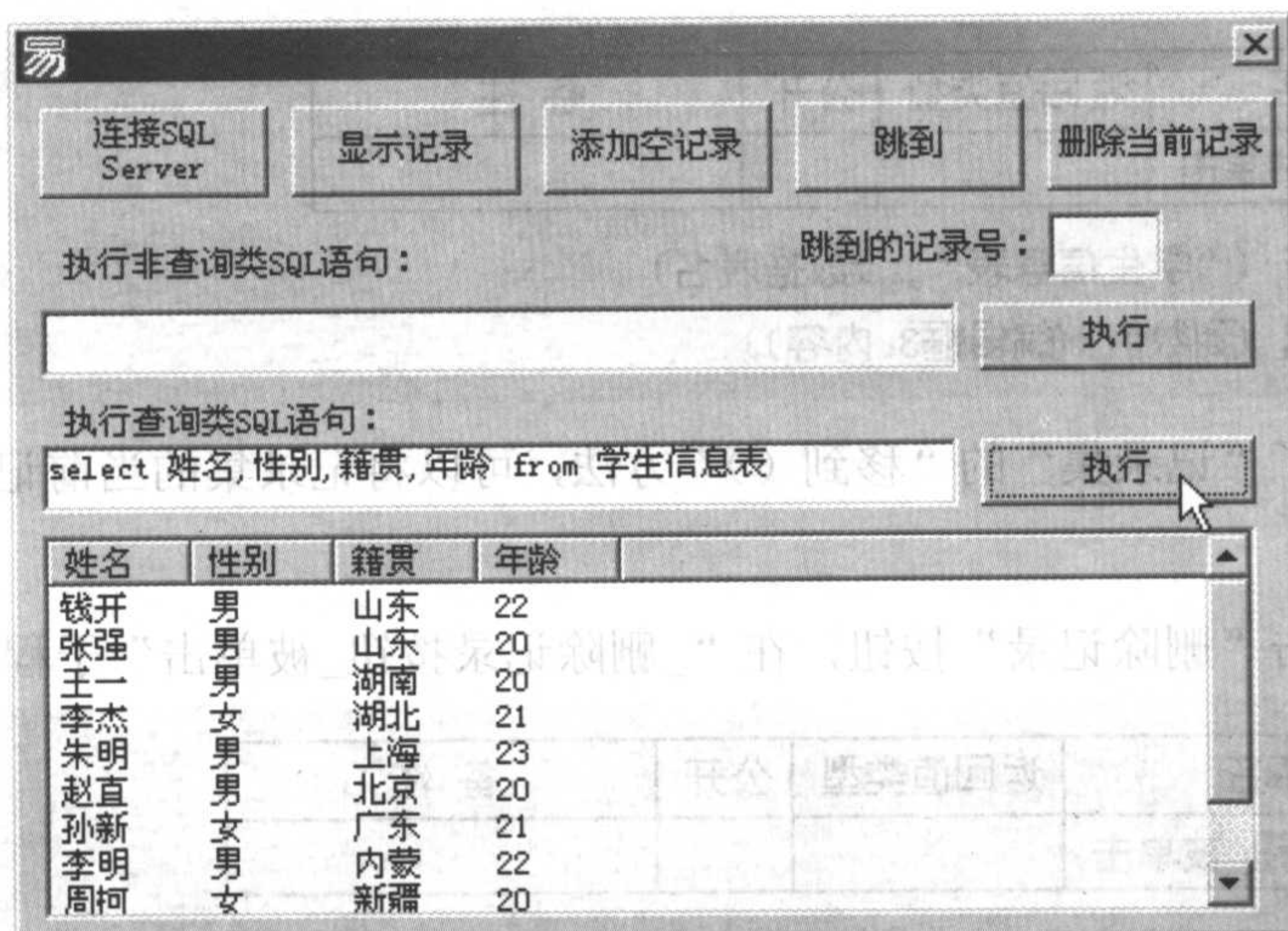


图 12-46 使用易程序操作 SQL Server 数据库

该例程参看随书光盘例程“操作 SQL Server 数据库.e”。

## 12.7 MYSQL 数据库

MYSQL 是一个多线程的，使用结构化查询语言(SQL)数据库服务器。MYSQL 的执行性能高，运行速度快，操作非常简单，并且还支持 Linux 操作系统。

MYSQL 的 Windows 环境控制界面如图 12-47 所示。

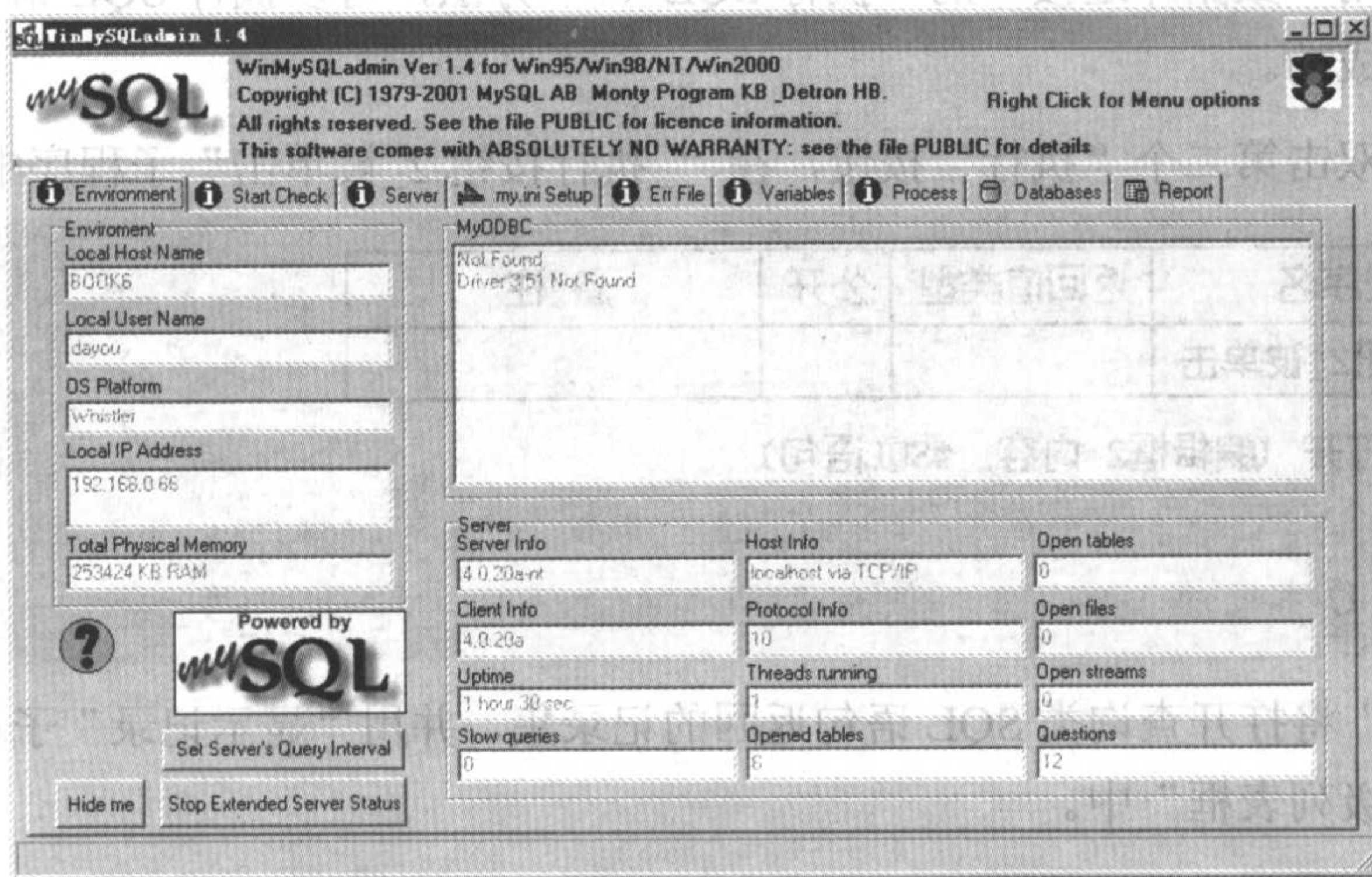


图 12-47 MYSQL 界面

数据库窗口如图 12-48 所示，看到有 test 数据。



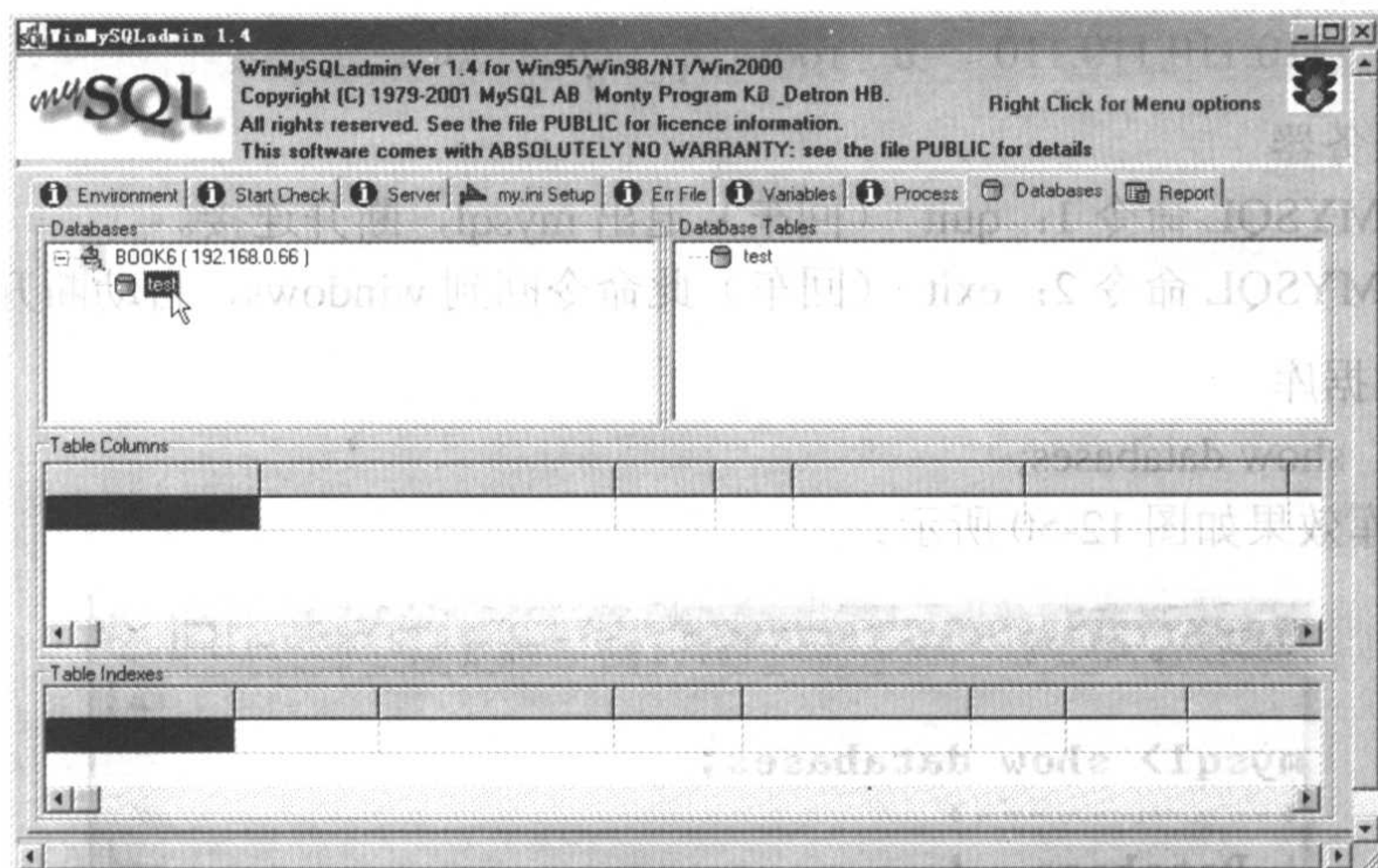


图 12-48 MYSQL 数据库界面

### 12.7.1 MYSQL 常用命令

#### 1. 首先连接服务器

大家以下以 Windows2000 以上系统作例子说明。MYSQL 操作皆为文本命令行模式，所以要做一下准备工作。

(1) 复制系统“system32”目录下的 cmd.exe 到 mysql 安装目录

(2) 运行 cmd.exe，出现命令提示符窗口

连接服务器格式：mysql -h 主机地址 -u 用户名 -p 用户密码

(3) 连接到本机上的 MYSQL

在命令提示符窗口，键入命令 **mysql -h localhost -u root -p**，回车后提示输入密码，如果刚安装好 MYSQL，超级用户 root 是没有密码的，直接回车即可登录到 MYSQL 中了，MYSQL 的提示符是：mysql>。如图 12-49 所示。

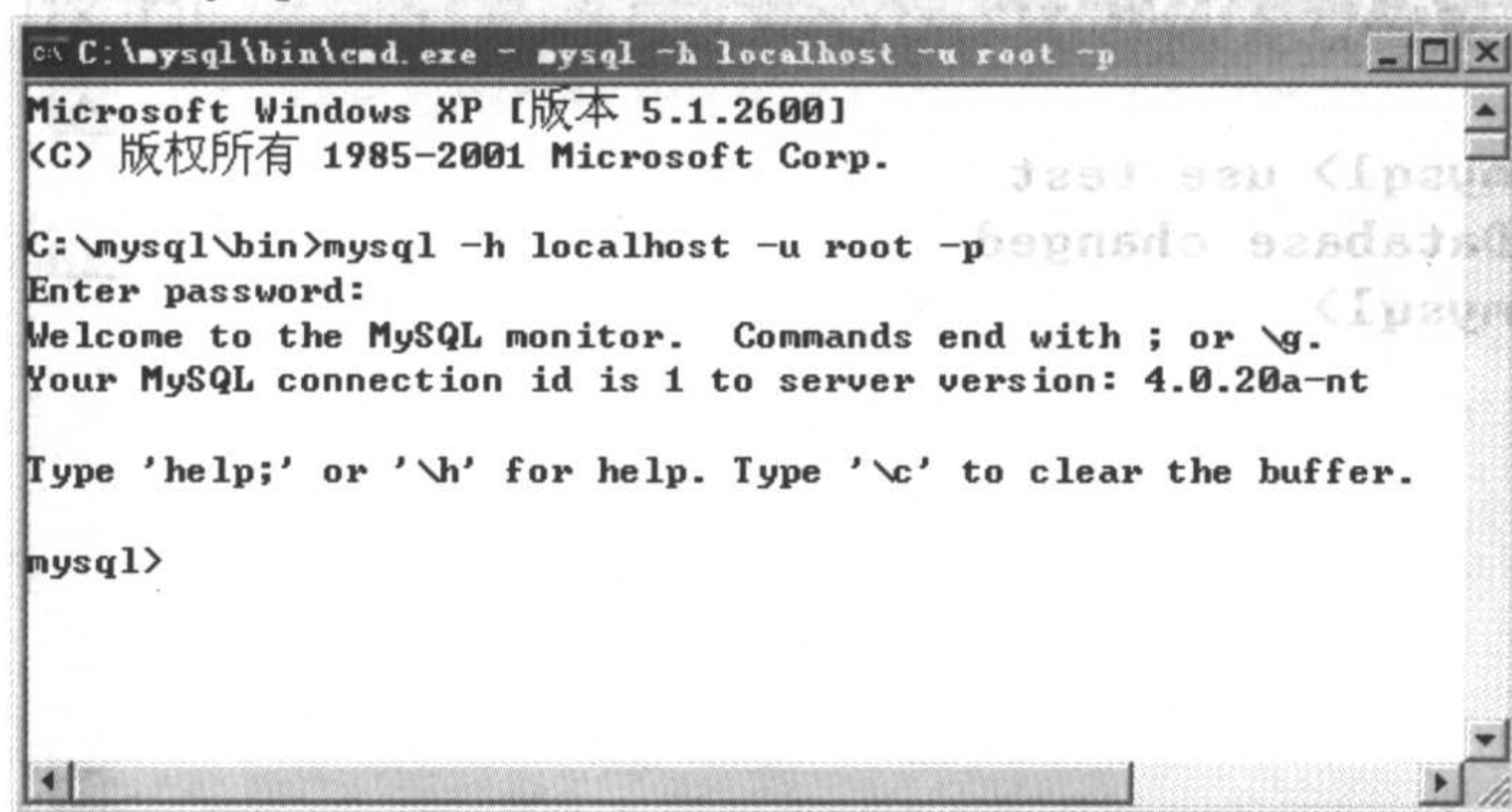


图 12-49 在 DOS 中连接到 MYSQL 服务器

(4) 连接到远程主机上的 MYSQL

假设远程主机的 IP 为：110.110.110.110，用户名为 root，密码为 abcd123，则键入以下命令：





```
mysql -h 110.110.110.110 -u root -p abcd123
```

## 2. 断开服务器

(1) 退出 MYSQL 命令 1: quit (回车) 退出 mysql, 断开连接。

(2) 退出 MYSQL 命令 2: exit (回车) 此命令回到 windows, 自动断开 mysql。

## 3. 查看数据库

命令格式: show databases;

查看数据库效果如图 12-50 所示。

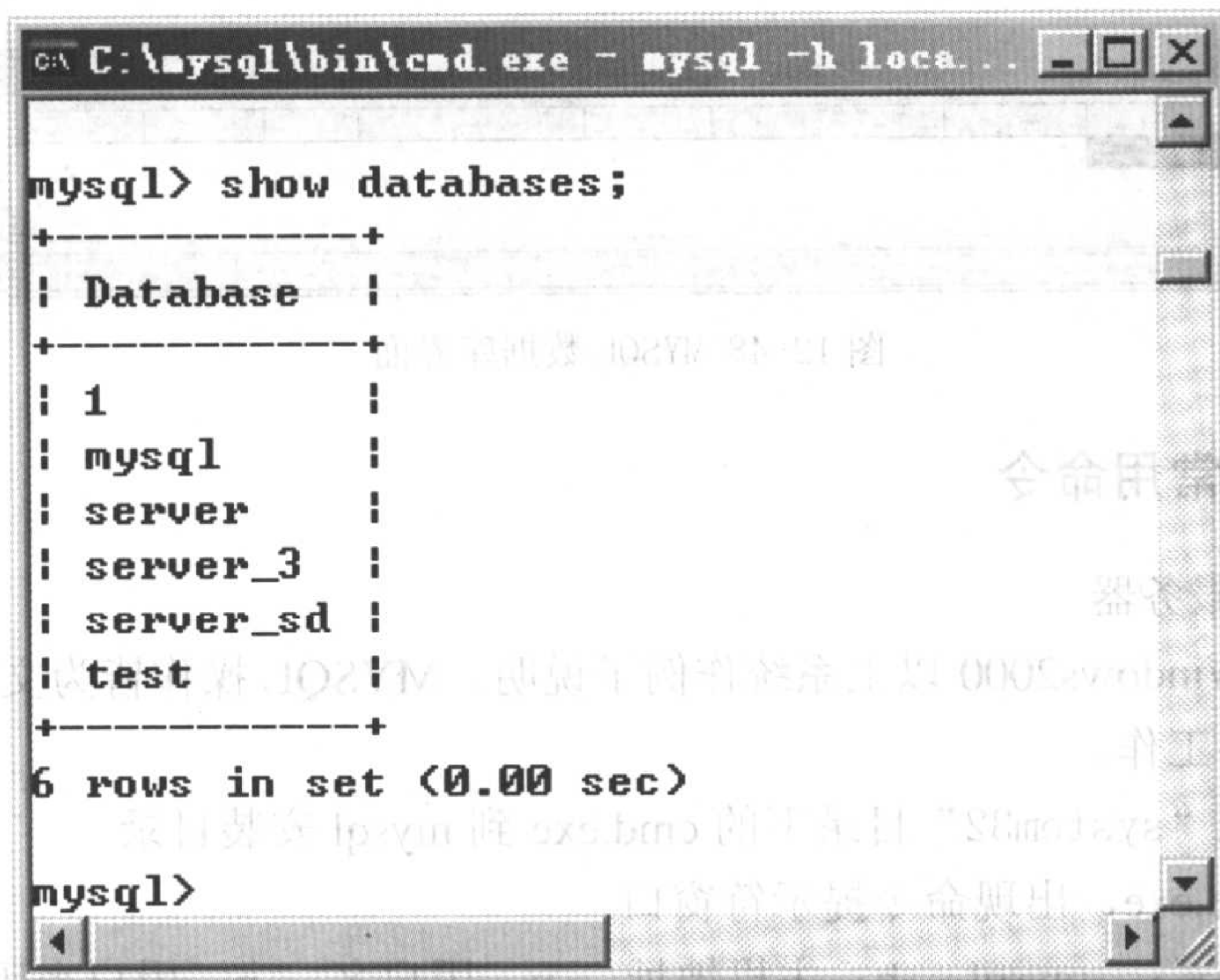


图 12-50 查看 MYSQL 数据库

## 4. 选择数据库

命令格式: use 数据库名, 键入命令后, 如图 12-51 所示。

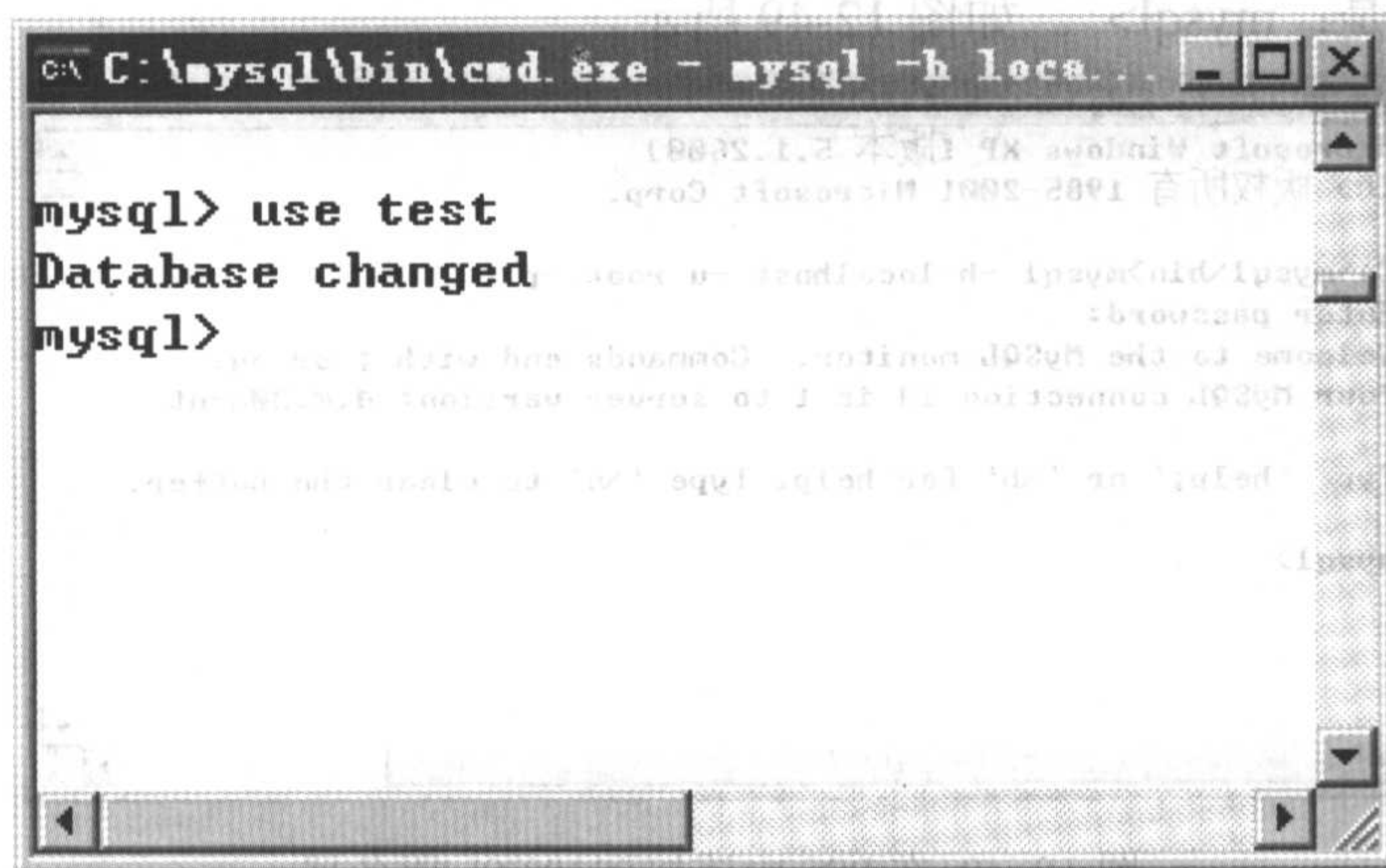


图 12-51 选择数据库

## 5. 查看表

命令格式: show tables, 查看表效果如图 12-52 所示。



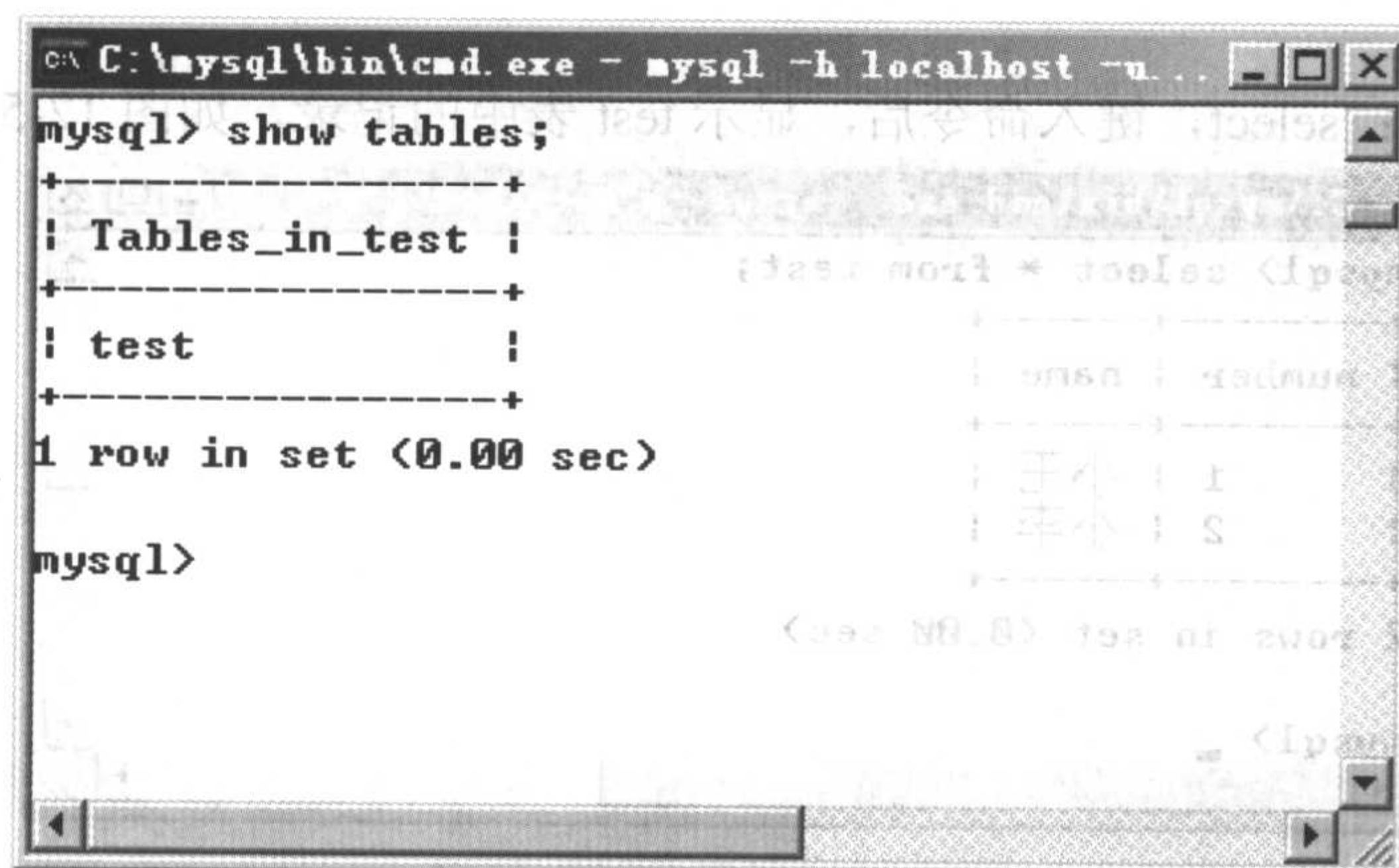


图 12-52 查看表

## 6. 查看表结构

命令格式: describe 表名, 查看表效果如图 12-53 所示。

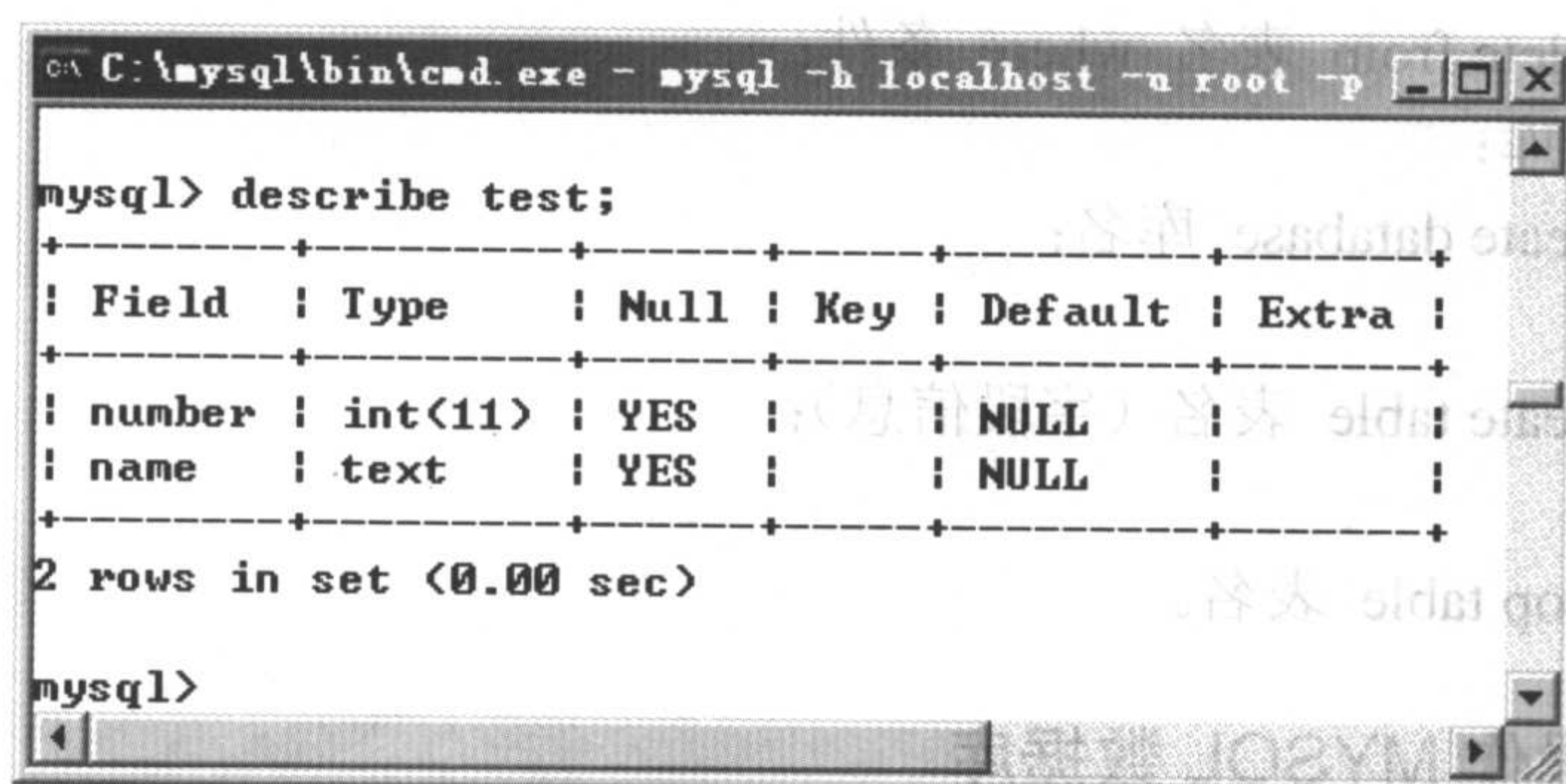


图 12-53 查看表结构

## 7. 插入记录

命令格式: insert into 表名 set 字段名=值, 键入命令后, 显示 test 表中的记录, 如图 12-54 所示。

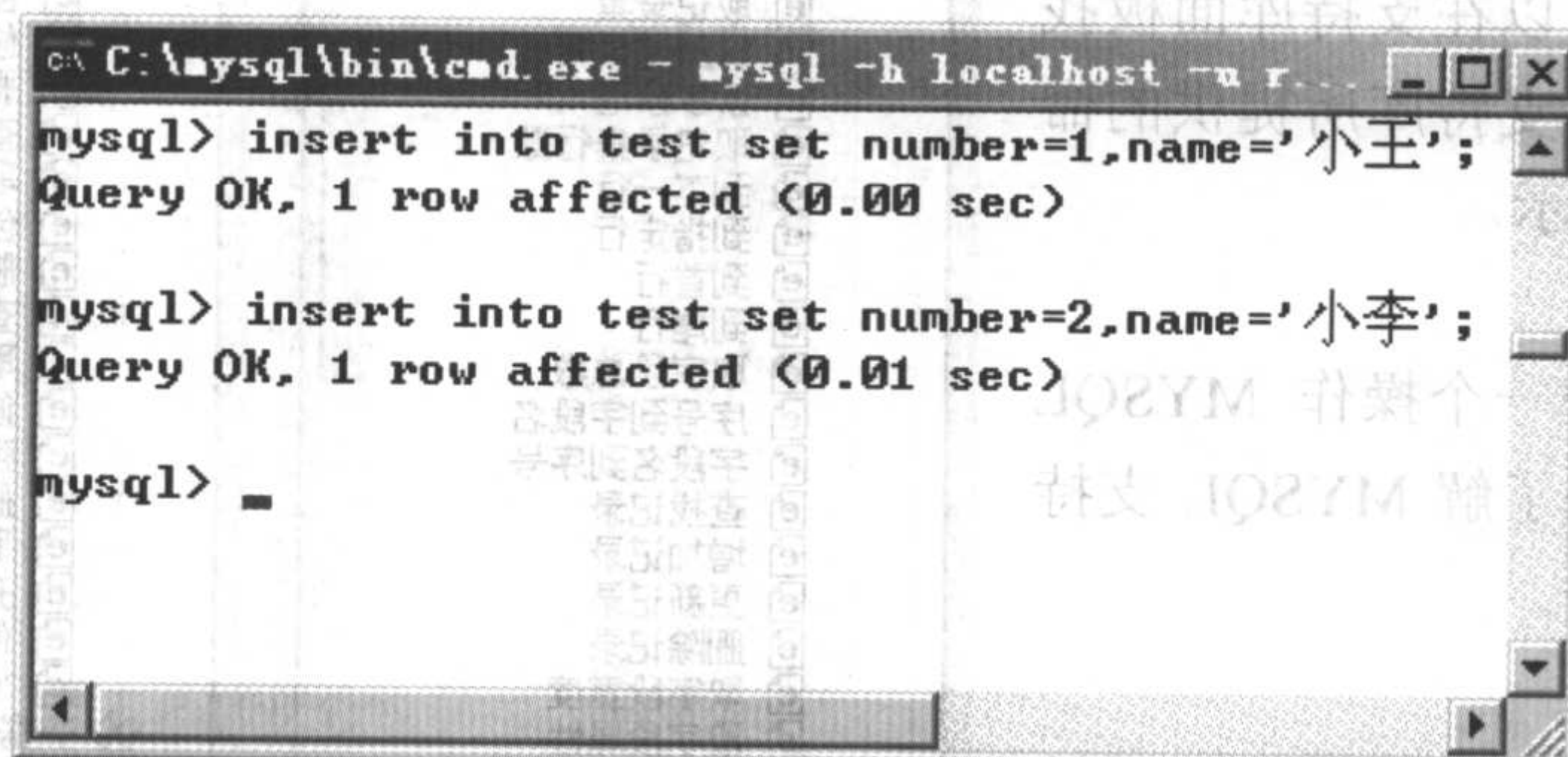


图 12-54 添加记录

## 8. 其他常用命令

用第一节介绍的常用 SQL 语句, 可以对 MYSQL 数据库进行操作, 例如常用的 SQL



语句:

(1) 查询语句: select, 键入命令后, 显示 test 表中的记录, 如图 12-55 所示。

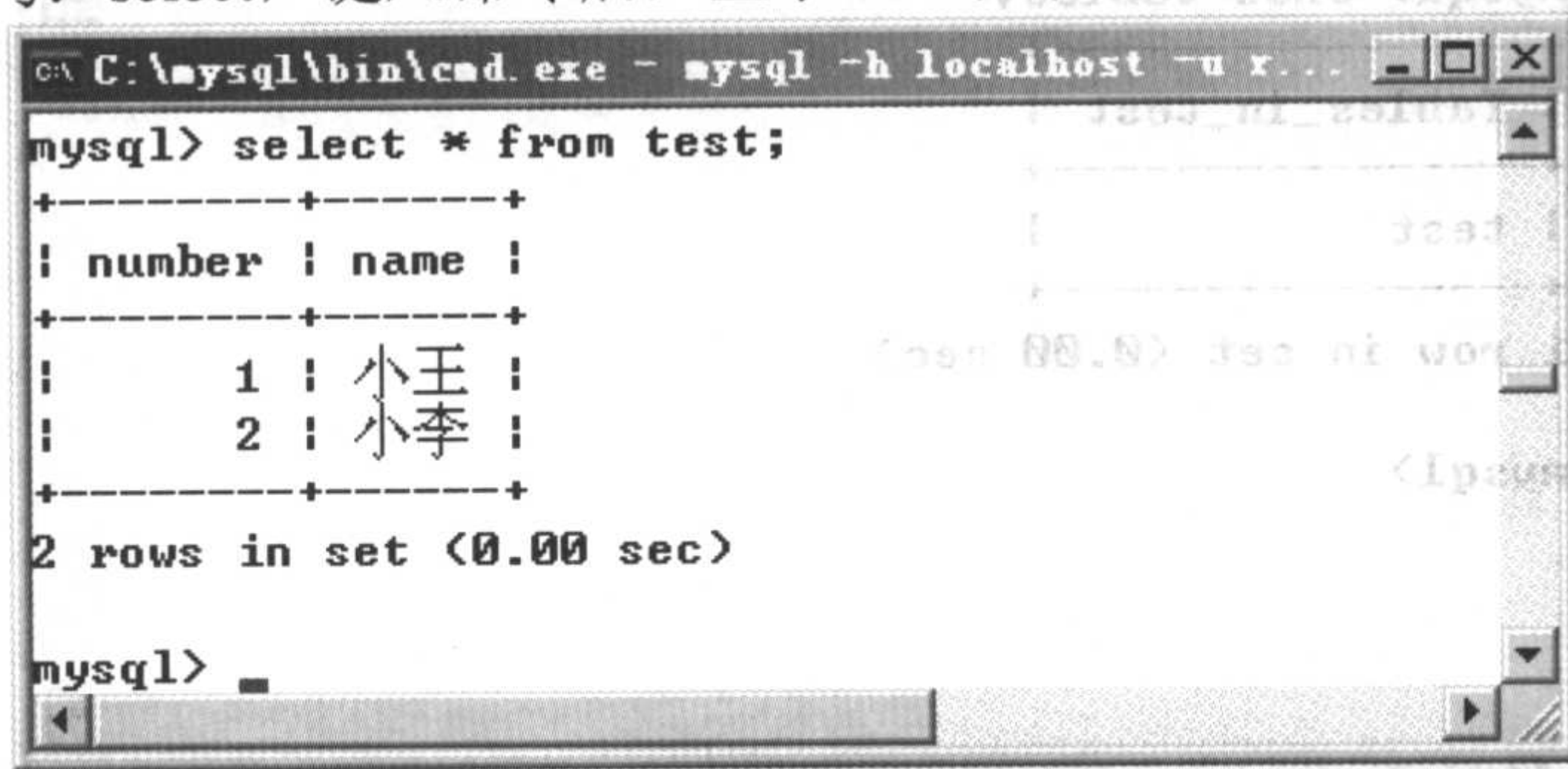


图 12-55 查看表中所有记录

(2) 修改记录:

命令格式: update 表名 set 更改值 where 条件;

(3) 删除记录:

命令格式: delete from 表名 where 条件;

(4) 创建数据库:

命令格式: create database 库名;

(5) 创建表:

命令格式: create table 表名 (字段信息);

(6) 删除表:

命令格式: drop table 表名。

## 12.7.2 易语言操作 MYSQL 数据库

易语言自 3.6 版推出以后, 提供了 MYSQL 支持库, 可以通过该支持库, 用易语言直接对 MYSQL 数据库进行操作, 大家可以在支持库面板找到该支持库和查看支持库所提供的命令, 如图 12-56 所示。

下面, 就结合一个操作 MYSQL 数据库的易程序来了解 MYSQL 支持库的一些常用命令。

第 1 步, 新建一个易程序, 设计程序界面如图 12-57 所示。

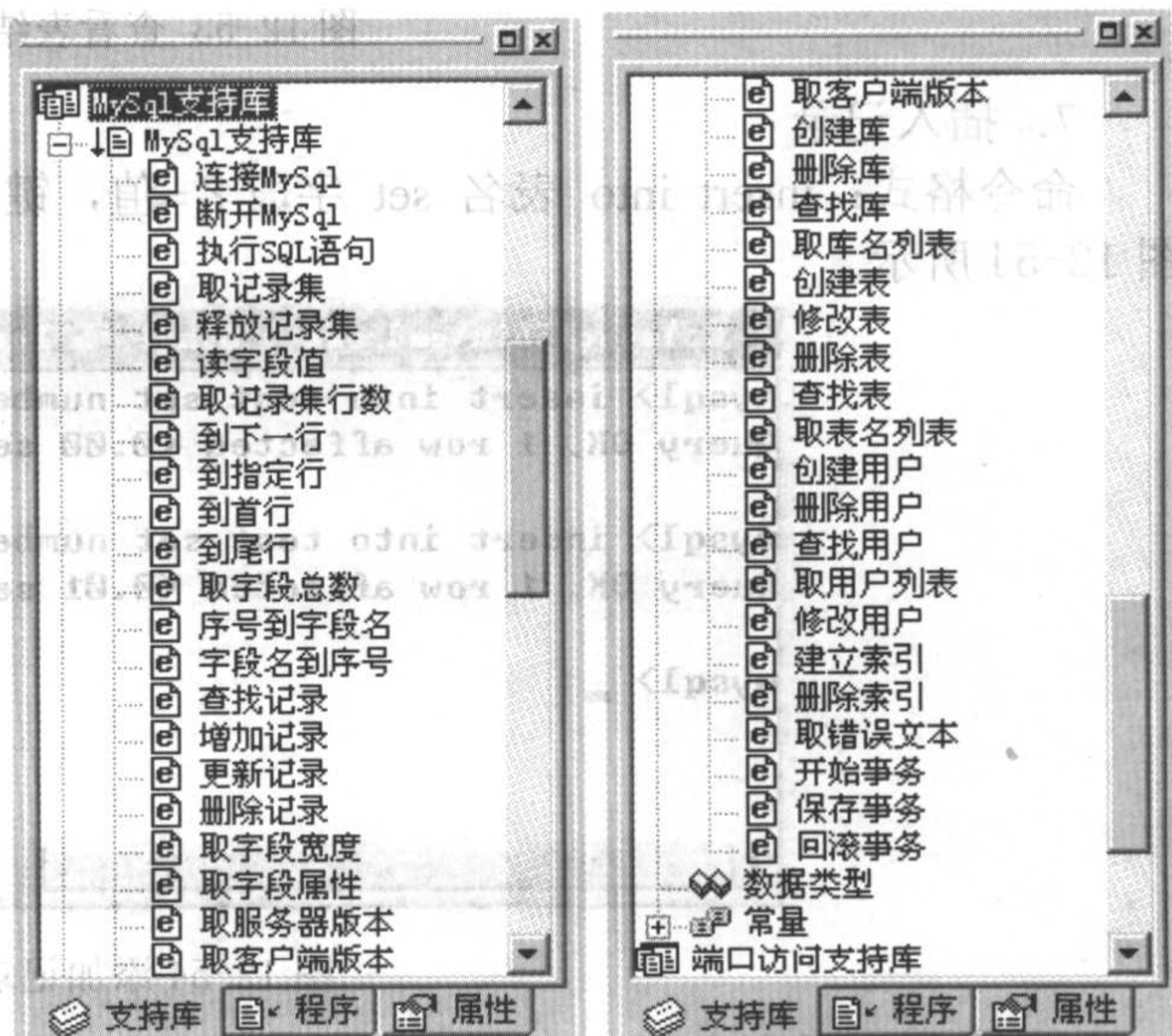


图 12-56 MYSQL 支持库命令



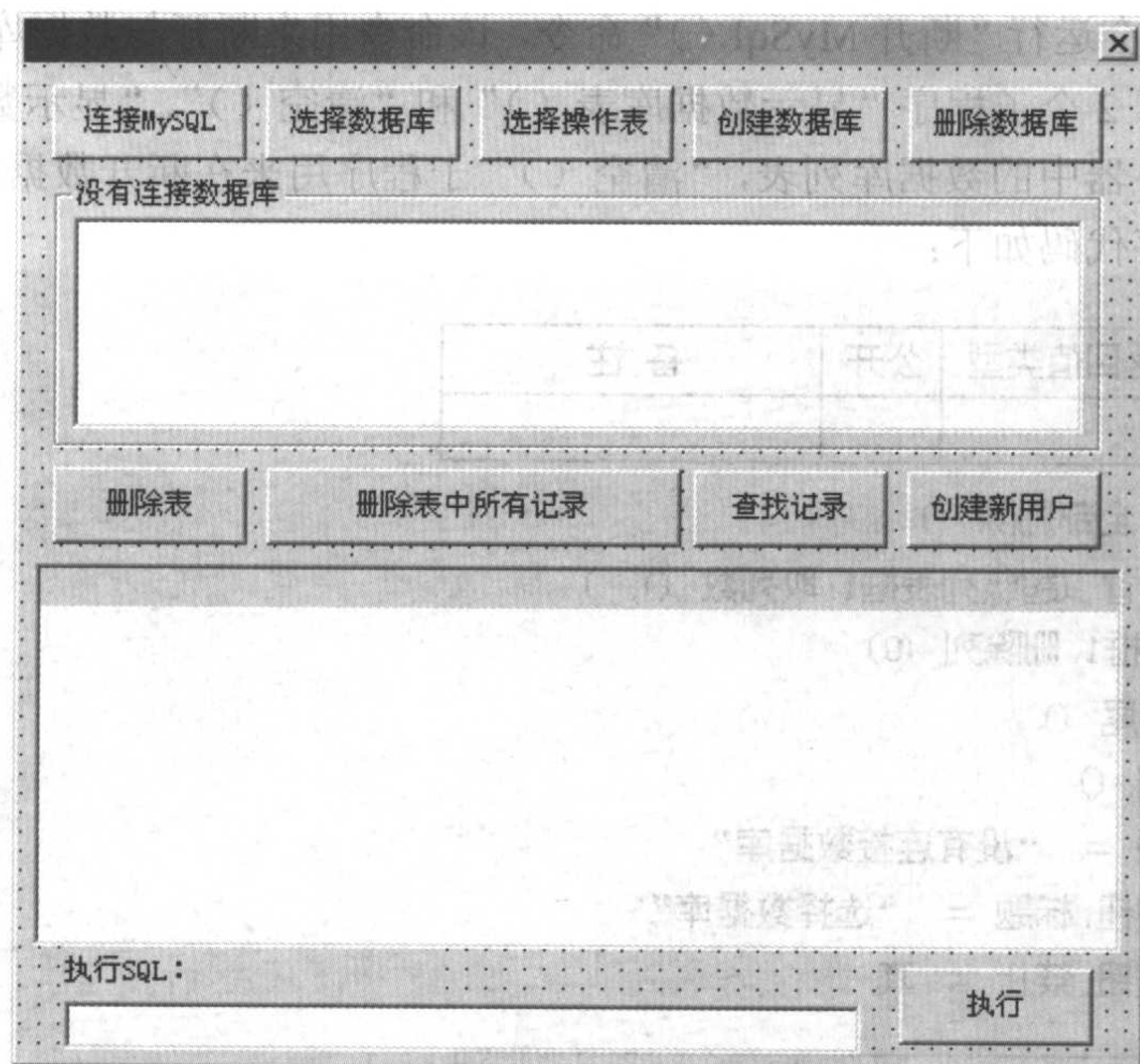


图 12-57 操作 MYSQL 数据库例程界面

例程中添加的按钮，都将“名称”属性改为和其功能对应的名称。

第 2 步，双击“连接 MYSQL”按钮，在“\_连接 MYSQL 按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_连接MySQL按钮_被单击			

```

--- 如果 (连接MySQL按钮.标题 = “连接MySQL”)
    MySQL句柄 = 连接MySQL (“”, “root”, “”, “”, 0)
    --- 如果 (MySQL句柄 ≠ 0)
        信息框 (“连接成功”, 0, )
        显示数据库表 ()
        --- 连接MySQL按钮.标题 = “断开服务器”
        信息框 (“连接失败”, 0, )
    --- 断开MySQL (MySQL句柄)
    连接MySQL按钮.标题 = “连接MySQL”
    清空 0
    
```

在操作 MYSQL 数据库前，首先要使用“连接 MySQL ()”命令，连接到指定的数据库，该命令运行后返回一个 MYSQL 句柄，后面对 MYSQL 数据库进行操作都要通过该句柄进行。所以这里创建了一个名为“MYSQL 句柄”的整数型程序集变量，程序中，在成功连接到服务器后就将“MYSQL”按钮的标题改为“断开服务器”，所以当按钮的标题为“断





开服务器”时，就会运行“断开 MySQL ()”命令，该命令用来断开与数据库服务器的连接。

程序中用到了 2 个子程序“显示数据库表 ()”和“清空 ()”。“显示数据库表 ()”子程序用来显示服务器中的数据库列表，“清空 ()”子程序用来在断开数据库后将程序恢复到起始状态。程序代码如下：

子程序名	返回值类型	公开	备注
清空			

超级列表框1.全部删除 ()

--- 计次循环首 (超级列表框1.取列数 (), )

超级列表框1.删除列 (0)

--- 计次循环尾 ()

列表框1.清空 ()

分组框1.标题 = “没有连接数据库”

选择数据库按钮.标题 = “选择数据库”

选择操作表按钮.禁止 = 真

子程序名	返回值类型	公开	备注
显示数据库表			

变量名	类型	静态	数组	备注
循环变量	整数型			
数据库名	文本型			

列表框1.清空 ()

--- 如果真 (取库名列表 (MySQL句柄))

记录集句柄 = 取记录集 (MySQL句柄)

--- 计次循环首 (取记录集行数 (记录集句柄), 循环变量)

读字段值 (记录集句柄, 0, 数据库名)

列表框1.加入项目 (数据库名, 循环变量)

到下一行 (记录集句柄)

--- 计次循环尾 ()

分组框1.标题 = “当前服务器数据库有” + 到文本 (取记录集行数 (记录集句柄)) + “个”

选择操作表按钮.禁止 = 真

超级列表框1.全部删除 ()

释放记录集 (记录集句柄)

“取库名列表 ()”命令，可以取出当前可以操作的所有数据库名。该命令返回的记录集要通过“取记录集 ()”命令取得。

“取记录集 ()”命令，可以取得当前返回的一个记录集的句柄，对记录集操作都要用到该句柄，所以程序中定义了一个名为“记录集句柄”的整数型程序集变量。

“读字段值 ()”命令，可以用来读取记录集中指定字段的值，程序中用了一个循环来将数据库表显示在列表框中。

“取记录集行数 ()”命令，可以取出记录集中的记录总数。“到下一行 ()”命令用来



将当前记录指针跳到下一记录。

在一个记录集不需要使用时，就要使用“释放记录集 ()”命令将其释放，否则将容易产生错误。

第 3 步，双击“选择数据库”按钮，在“\_选择数据库按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_选择数据库按钮_被单击			

```

--- 如果 (选择数据库按钮.标题 = “选择数据库”)
    --- 如果 (列表框1.现行选中项 ≠ -1)
        MySQL句柄 = 连接MySQL (“”, “root”, “”, 列表框1.取项目文本
            (列表框1.现行选中项), 0)
        ▶ 信息框 (“请选择数据库”, 0, )
        返回 ()
        ▶ 如果 (MySQL句柄 ≠ 0)
            显示表名 ()
            选择操作表按钮.禁止 = 假
            ▶ 信息框 (“选择数据库失败”, 0, )
        选择数据库按钮.标题 = “显示数据库列表”
        ▶ 显示数据库表 ()
        选择数据库按钮.标题 = “选择数据库”
    
```

上述程序中，当选择了一个数据库后，就重新连接 MySQL 数据库，并将当前操作的数据库设为列表框中选择的数据库。

程序还用到了一个子程序“显示表名 ()”，该子程序中用来显示被选择的数据库中有多少个表。程序代码如下：

子程序名	返回值类型	公开	备注
显示表名			

变量名	类型	静态	数组	备注
循环变量	整数型			
表名	文本型			

```

列表框1.清空 ()
--- 如果真 (取表名列表 (MySQL句柄))
    记录集句柄 = 取记录集 (MySQL句柄)
    ▶ 计次循环首 (取记录集行数 (记录集句柄), 循环变量)
        读字段值 (记录集句柄, 0, 表名)
        列表框1.加入项目 (表名, 循环变量)
        到下一行 (记录集句柄)
    --- 计次循环尾 ()
    分组框1.标题 = “当前数据库有表” + 到文本 (取记录集行数 (记录集句柄)) + “个”
    释放记录集 (记录集句柄)
    
```





上述程序中，使用了“取表名列表（）”命令，可以取出当前数据库中所有的表名，返回的记录集要用“取记录集（）”命令取得。

第4步，双击“选择操作表”按钮，在“\_选择操作表按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_选择操作表按钮_被单击			

```

当前操作表 = 列表框1.取项目文本 (列表框1.现行选中项)
释放记录集 (记录集句柄)
执行SQL语句 (MySQL句柄, "select * from " + 当前操作表)
记录集句柄 = 取记录集 (MySQL句柄)
显示记录 ()
  
```

上述程序中，使用了一个文本型程序集变量“当前操作表”，用来存放当前被操作的表名。

“执行SQL语句”命令，可以执行一段“SQL”语句，如果执行查询语句，返回的记录集要通过“取记录集（）”命令取得。

程序中调用了“显示记录（）”子程序，该子程序用来将当前记录集中的记录显示在超级列表框中，代码如下：

子程序名	返回值类型	公开	备注
显示记录			

变量名	类型	静态	数组	备注
循环变量2	整数型			
字段名变量	文本型			
循环变量1	整数型			
读取字段号	整数型			

超级列表框1.全部删除 ()

--- 计次循环首 (超级列表框1.取列数 (), )

超级列表框1.删除列 ()

--- 计次循环尾 ()

--- 计次循环首 (取字段总数 (记录集句柄), 循环变量1)

序号到字段名 (记录集句柄, 循环变量1 - 1, 字段名变量)

超级列表框1.插入列 (-1, 字段名变量, , , )

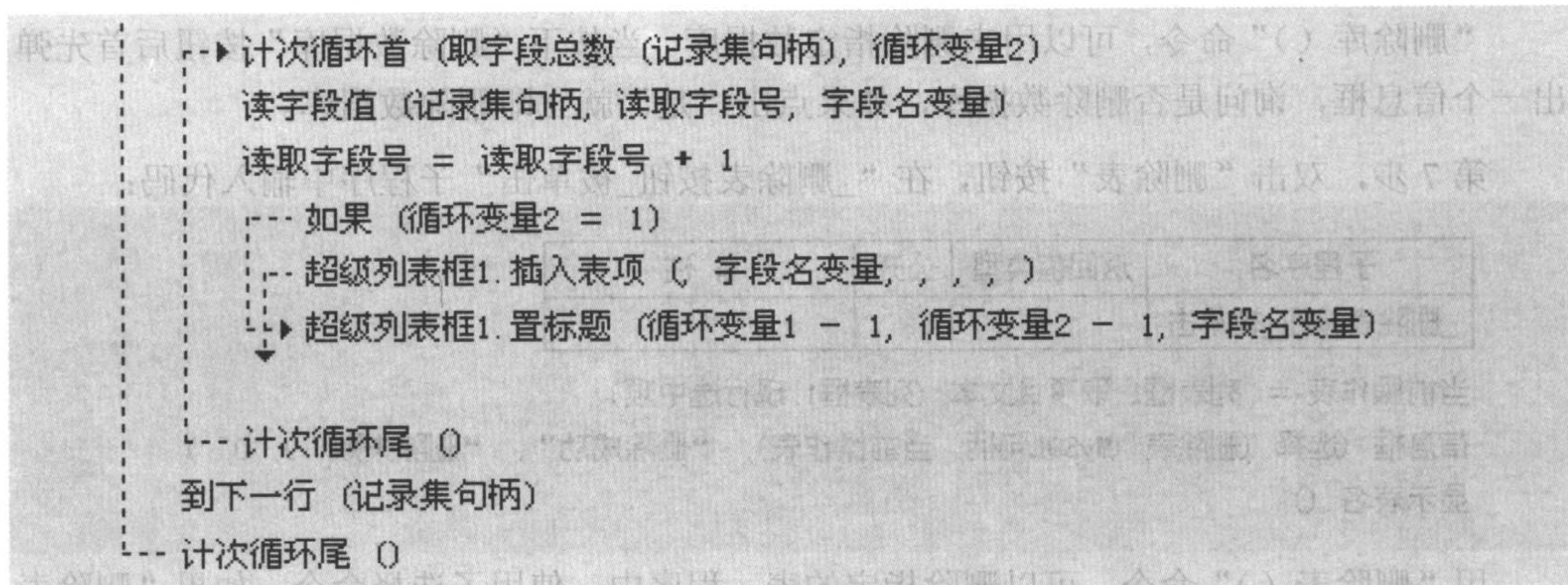
--- 计次循环尾 ()

到首行 (记录集句柄)

--- 计次循环首 (取记录集行数 (记录集句柄), 循环变量1)

读取字段号 = 0





由于记录集中的字段数量不固定，所以子程序中首先将超级列表框的所有列删除，然后重新加入新列。

“到首行 ()” 命令，将当前记录指针跳到记录集的首记录位置。

“取字段总数 ()” 命令，取出当前记录集中的字段总数。

第 5 步，双击“创建数据库”按钮，在“\_创建数据库按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_创建数据库按钮_被单击			

变量名	类型	静态	数组	备注
数据库名	文本型			

输入框 ( “请输入要创建的数据库名：”, , 数据库名, #输入文本)  
 创建库 (MySQL句柄, 数据库名)  
 显示数据库表 ()  
 选择数据库按钮.标题 = “选择数据库”

上述程序实现了当按下“创建数据库”按钮后，就弹出一个输入框，用输入框来输入欲创建的数据库名。“创建库 ()” 命令，可以在已经连接的 MYSQL 服务器中创建一个新数据库。

第 6 步，双击“删除数据库”按钮，在“\_删除数据库按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_删除数据库按钮_被单击			

```

    -- 如果真 (列表框1.现行选中项 ≠ -1)
    -- 如果 (信息框 (“确定要删除该数据库?”, #询问图标 + #是否钮, ) = #是钮)
    删除库 (MySQL句柄, 列表框1.取项目文本 (列表框1.现行选中项))
    显示数据库表 ()
    返回 0
  
```





“删除库 ()”命令，可以用来删除指定数据库。当按下“删除数据库”按钮后首先弹出一个信息框，询问是否删除数据库，如果点击“是”就立即删除数据库。

第7步，双击“删除表”按钮，在“\_删除表按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_删除表按钮_被单击			

当前操作表 = 列表框1.取项目文本 (列表框1.现行选中项)

信息框 (选择 (删除表 (MySQL句柄, 当前操作表), “删除成功”, “删除失败”), 0, )

显示表名 0

用“删除表 ()”命令，可以删除指定的表，程序中，使用了选择命令，如果“删除表 ()”命令返回真，即成功删除表，信息框就显示“删除成功”否则显示“删除失败”。

第8步，双击“删除表中所有记录”按钮，在“\_删除记录按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_删除记录按钮_被单击			

--- 如果真 (信息框 (“是否确定删除?”, #询问图标 + #是否钮, ) = #是钮)

删除记录 (MySQL句柄, 当前操作表, “”)

释放记录集 (记录集句柄)

显示记录 0

程序中，使用了“删除记录 ()”命令，该命令第1个参数填写 MySQL 句柄，第2个参数填写欲操作的表名，第3个参数是删除符合条件的记录，如果为空，则删除所有记录，也可以给该参数一个条件表达式，可以删除符合条件的指定记录，例如删除“number”字段等于“2”的记录，使用代码：

删除记录 (MYSQL 句柄, 当前操作表, number=2)

第9步，插入一个新窗口，用来查找指定记录，将窗口名称定义为“查找窗口”，查找窗口如图 12-58 所示：

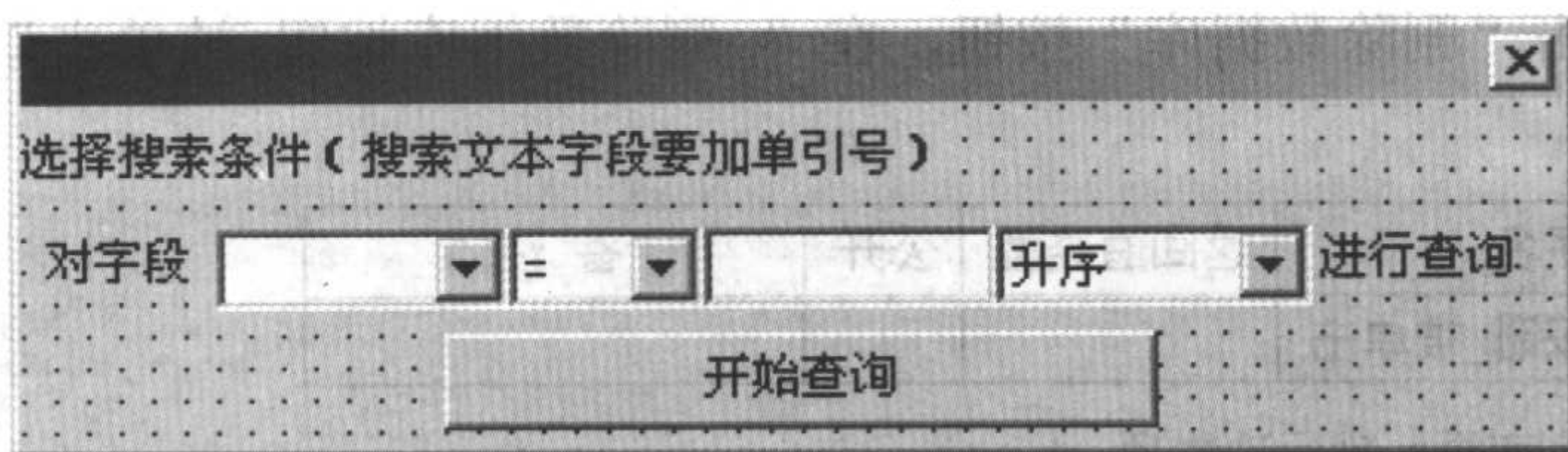


图 12-58 查找窗口

第1个组合框显示当前表中的所有字段名，第2个组合框中加入所有的运算符，编辑框用来填写查找的内容。

首先双击“查找窗口”在“\_查找窗口\_创建完毕”子程序中输入代码：



子程序名	返回值类型	公开	备注
_查找窗口_创建完毕			

变量名	类型	静态	数组	备注
循环变量	整数型			
字段名变量	文本型			

---▶ 计次循环首 (取字段总数 (\_启动窗口.记录集句柄), 循环变量)  
 序号到字段名 (\_启动窗口.记录集句柄, 循环变量 - 1, 字段名变量)  
 组合框1.加入项目 (字段名变量, )  
 --- 计次循环尾 0

程序中使用了 1 个计次循环, 将记录集中的所有字段名填加到组合框中。

然后双击“查找窗口”中的“开始查询”按钮, 在“\_开始查询按钮\_被单击”子程序中输入代码:

子程序名	返回值类型	公开	备注
_开始查询按钮_被单击			

释放记录集 (\_启动窗口.记录集句柄)

\_启动窗口.记录集句柄 = 查找记录 (\_启动窗口.MySQL句柄, \_启动窗口.当前操作表, “\*”, 组合框1.取项目文本 (组合框1.现行选中项) + 组合框2.取项目文本 (组合框2.现行选中项) + 编辑框1.内容, 组合框1.取项目文本 (组合框1.现行选中项) + “ ” + 多项选择 (组合框1.现行选中项 + 1, “ASC”, “DESC”))

销毁 0

程序中使用了“查找记录 ()”命令, 命令的格式如下:

查找记录 (MySQL 句柄, 表名, 字段名, 查找条件, 排序条件)

该命令返回值是一个记录集句柄, 所以运行该命令后不需要使用“取记录集 ()”命令来取得记录集句柄。

程序中将查找的结果存放在“\_启动窗口”程序集中的“记录集句柄”变量中。

第 10 步, 回到“\_启动窗口”, 双击“查找记录”按钮, 在“\_查找记录按钮\_被单击”子程序中输入代码:

子程序名	返回值类型	公开	备注
_查找记录按钮_被单击			

载入 (查找窗口, , 真)

显示记录 0

第 11 步, 双击“创建用户”按钮, 在“\_创建新用户按钮\_被单击”子程序中输入代码:





子程序名	返回值类型	公开	备注
_创建新用户按钮_被单击			

变量名	类型	静态	数组	备注
用户名	文本型			
密码	文本型			

输入框 (“输入用户名”, , , 用户名, )

输入框 (“输入密码”, , , 密码, )

信息框 (选择 (创建用户 (MySQL句柄, “”, 用户名, 密码, “\*”, “\*”, #所有权限), “创建成功”, “创建失败”), 0, )

程序中使用了“创建用户 ()”命令来创建一个新用户, 并使用了2个输入框来输入创建用户的用户名及密码。

“创建用户 ()”命令的格式为:

创建用户 (MySQL句柄, 主机, 用户名, 密码, 库名, 表名, 权限)

可以根据需要规定创建的用户可以操作的数据库及表。

第12步, 双击执行按钮, 在“\_执行按钮\_被单击”子程序中输入代码:

子程序名	返回值类型	公开	备注
_执行按钮_被单击			

释放记录集 (记录集句柄)

执行SQL语句 (MySQL句柄, 编辑框1.内容)

记录集句柄 = 取记录集 (MySQL句柄)

显示记录 0

最后试运行程序, 查看运行效果, 如图12-59所示。



图12-59 “MySQL使用例程.e”运行效果



该例程参见随书光盘中例程“MySQL 使用例程.e”。

该例程中只使用了部分 MYSQL 支持库中的常用命令进行介绍，其他常用命令可以根据命令帮助来使用，这里就不一一介绍了。

## 12.9 本章小结

数据库是大家编程中使用最多的工具，易语言支持国际通用的 ODBC 和 ADO 数据库访问协议，因此所有的外部大型数据库，如 MYSQL、Sql Server、Oracle、Sybase 等数据库都可以被易语言程序所访问。通过本章的学习，大家可以大体了解 ODBC 与 ADO 的操作方式，

由于 MYSQL 数据库为开放源代码数据库系统，较为安全，并且功能强大，因此在易语言中对 MYSQL 数据库进行了直接的访问支持（不通过任何数据库协议），速度较快，易于安装与发布，是大家理想的数据库开发工具，并且由于它也支持 Linux 操作系统平台，因此成为跨平台的数据库首选工具。

大家也可以进行以下练习：

1. 练习 ODBC 与 ADO 两种对数据库的操作方式。
2. 跟着本书练习 Access 数据库的建立与操作。
3. 跟着本书练习安装 SQL Server 2000，及用易语言编写程序对它进行操作。
4. 跟着本书练习安装 MYSQL，及用易语言编写程序对它进行操作。
5. 试写基于 Linux 操作系统的 MYSQL 程序。





## 第十三章 API 的调用

### 13.1 API 简介

Windows 是一个多任务操作系统，除了协调应用程序的执行、分配内存和管理系统资源之外，它同时也可以看成是一个很大的服务中心。通过调用这个服务中心的各种服务，可以帮助应用程序实现开启窗口、描绘图形、使用周边设备等功能。调用这些服务的对象称作应用程序(Application)，实现调用的接口函数称之为应用程序接口(Application Programming Interface)，简称“API 函数”。

API 函数是 Windows 提供给应用程序与操作系统的接口，编程人员可以在不同的编程语言环境中对 API 函数进行调用，以编制出适合 Windows9x/WindowsNT/WindowsXP 操作系统的各类应用程序。本书主要讲解“WIN32 API”，也就是 Microsoft Windows 32 位平台中 API 函数接口中的应用。

追溯 API 的发展历程，当 Windows 操作系统开始占据主导地位的时候，开发 Windows 平台下的应用程序便成为主流。当时的 Windows 程序开发是一个比较复杂的工作，所能使用的应用程序接口唯有 API 函数，所以程序员必须熟记许多常用的 API 函数，还要对 Windows 操作系统有深入的了解。随着软件技术的不断发展，众多软件公司开发出多种运行于 Windows 操作平台上的优秀的可视化编程操作环境（诸如易语言、VB、Delphi 等）。这些可视化编程环境操作简单，界面友好，并借由组件和类库封装了许多常用的 API 函数，并赋予其简单且方便的操作方法。有了这些组件和类库，编程人员便可以把主要精力放在程序整体功能的设计上，而不必过多关注底层技术方面的细节，所以极大的提高了 Windows 应用程序的开发速度。

虽然组件和类库使应用程序的开发简单的多，但它们只能提供 Windows 的一般功能操作，对于比较复杂和特殊的功能来说，使用现有的组件和类库可能就无法实现，这时必然要涉及直接调用 API 函数。

对于 API 函数的调用，编程人员只需了解函数的意义与参数列表，然后把它们按正确的编程规则放到指定的程序中就可以了。在调用函数时，所定义参数的数据类型应该与实际的 DLL 库命令中的参数数据类型一致，否则程序将无法正常运行。

### 13.2 API 的定义

在可视化编程领域中，易语言也同样支持 API 函数的调用。大家首先一同来了解一下，



在易语言中是如何定义 API 函数的。

在 Windows 中，API 函数位于 DLL（动态链接库）文件中。在易语言里为了便于编程人员的理解，把调用 API 函数的命令称之为“DLL 命令”。

在“程序”面板中双击“DLL 命令”选项，在代码设计工作区中单击鼠标右键，从弹出的菜单中选择“新 DLL 命令”菜单项。如图 13-1 所示。

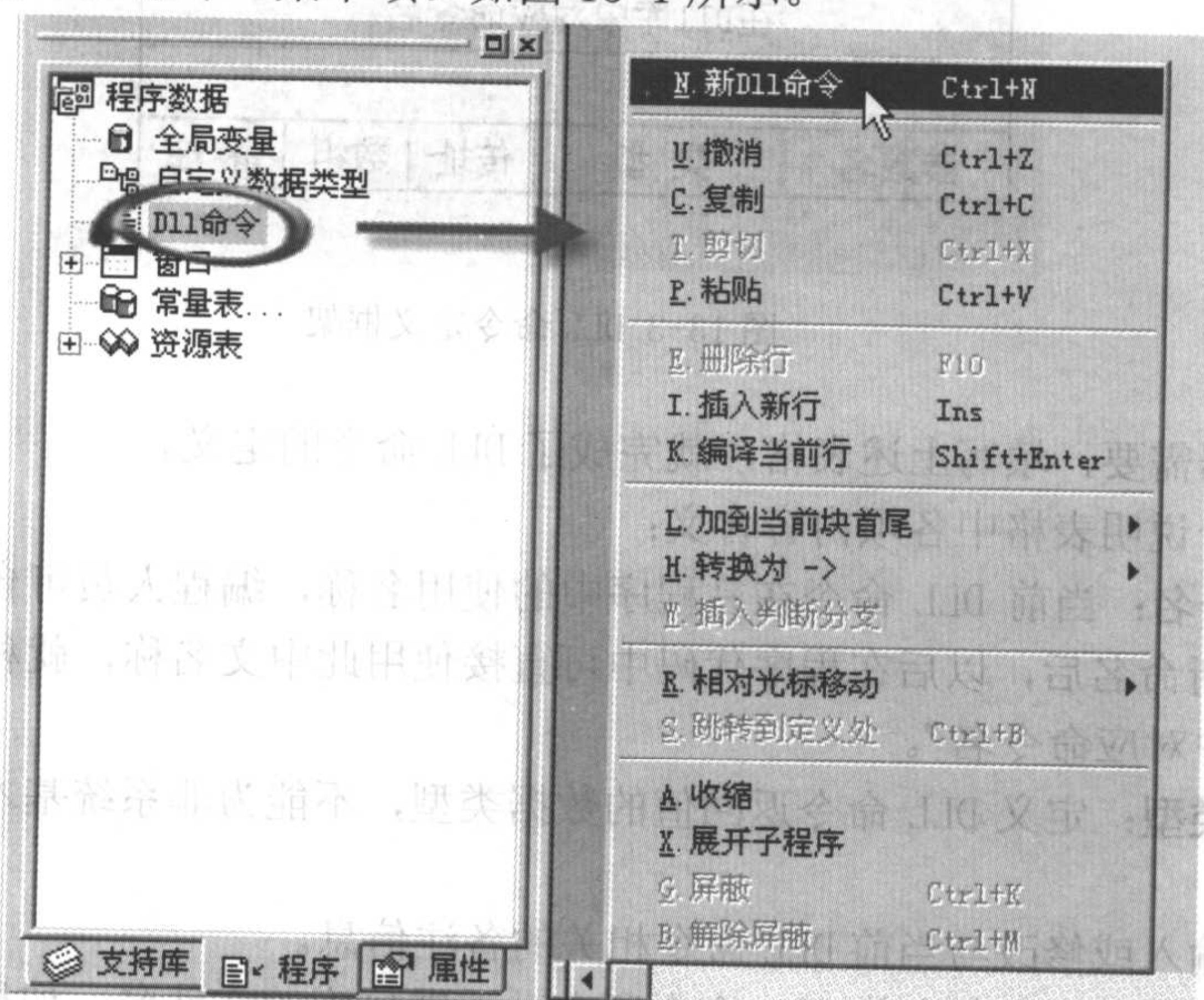


图 13-1 新建 DLL 命令

或者在双击“DLL 命令”选项之后，选择“插入”菜单中的“现行单元”或按下 Ctrl+N 快捷键，同样可以新建一个 DLL 命令。如图 13-2 所示。

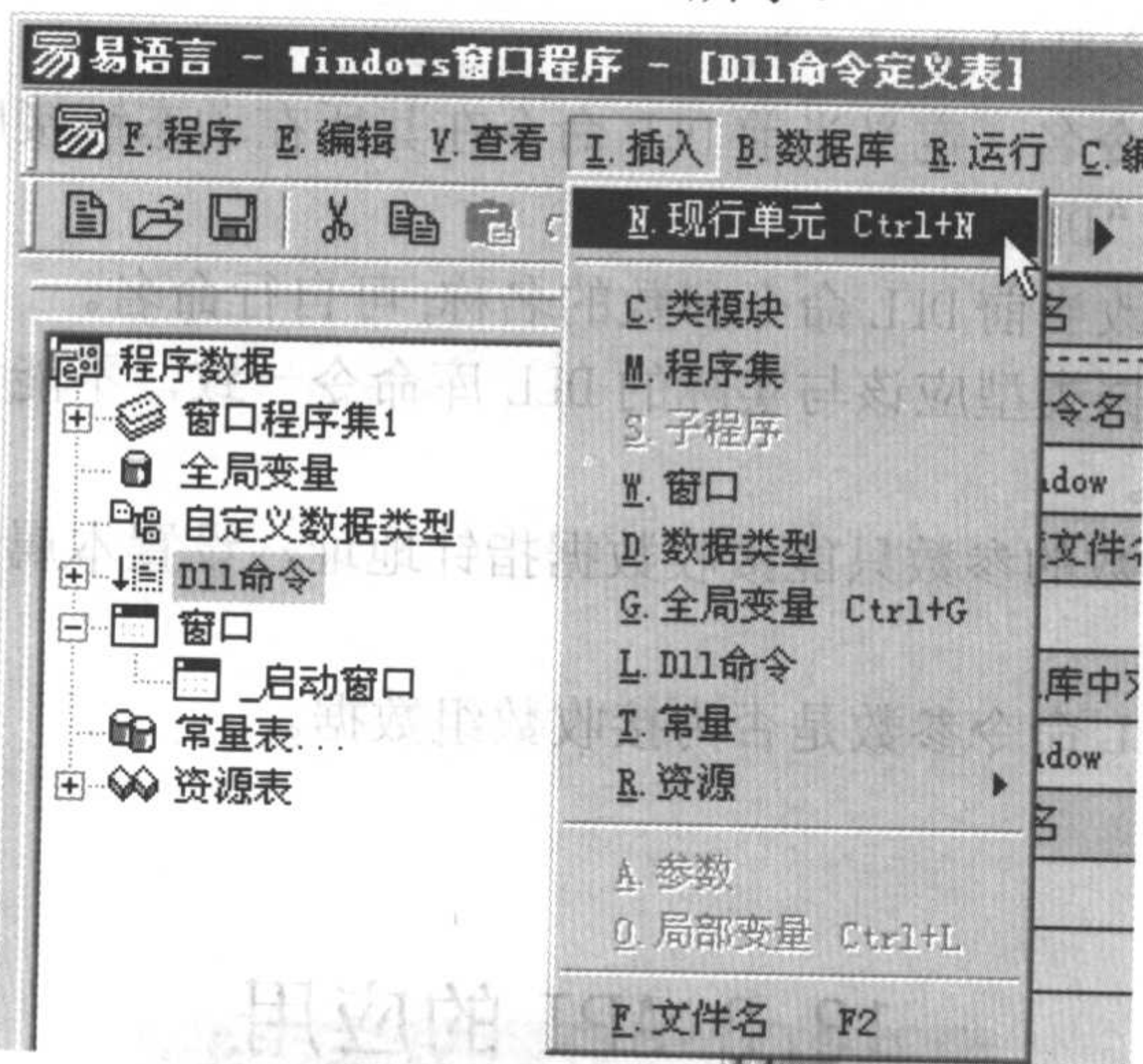


图 13-2 从“插入”菜单的“现行单元”进行 DLL 新建

其后，会在代码设计工作区中出现一个 DLL 命令的定义框架，如图 13-3 所示：





Dll命令名	返回值类型	备 注		
DLL命令1				
Dll库文件名				
在Dll库中对应命令名				
参数名	类 型	传址	数组	备 注

图 13-3 DLL 命令定义框架

只要根据需要，填写上述表格，就完成了 DLL 命令的定义。

下面逐一说明表格中各项内容含义：

**DLL 命令名：**当前 DLL 命令在易程序中的使用名称，编程人员可自行定义。使用中文为此 DLL 重新命名后，以后在程序代码中可直接使用此中文名称，就相当于使用了对应的“在 DLL 库中对应命令名”。

**返回值类型：**定义 DLL 命令返回值的数据类型，不能为非系统基本数据类型或字节集型。

**备注：**输入或修改与当前 DLL 命令相关的备注信息。

**DLL 库文件名：**定义当前 DLL 命令所在动态链接库的文件名。如果不指定库文件名，系统将默认在 Kernel32.dll、Gdi32.dll、User32.dll、Mpr.dll、Advapi32.dll、Shell32.dll 等 Windows 系统的基本应用程序界面函数（API）库中搜寻指定命令。**注意：**这里不要包含 DLL 库文件名的路径，程序运行后会自动定位该 DLL 库（查找顺序为：EXE 所在目录，当前目录，系统目录，系统安装目录……）。

**在 DLL 库中对应命令名：**定义当前 DLL 命令在其所在动态链接库中的名称，如果不指定，系统将默认等同于“DLL 命令名”。

**参数名：**输入或修改当前 DLL 命令参数的名称，可自行命名。

**类型：**所定义的数据类型应该与实际的 DLL 库命令一致，不能为复合数据类型数组、窗口、窗口单元、菜单。

**传址：**某些 API 函数的参数只能接收数据指针地址，设置本属性为真可实现参数数据的指针地址传递。

**数组：**设置当前 DLL 命令参数是否为接收数组数据。

### 13.3 API 的应用

Windows 系统标准 API 函数被编译到扩展名为 DLL 的文件。在 Windows9x 中，DLL 文件



一般被存放在系统目录下的 Windows\System 目录中,在 Windows2000 以上系统中存放在 WinNT\System32 目录中。在电脑技术突飞猛进的今天,很多软件公司为 Windows 系统开发了许多外部 API 函数。编程人员在开发过程中,既可以调用 Windows 系统自带的 API 函数,也可以调用 Windows 系统外部扩展 API 函数。

了解“DLL 命令”的一些基本规则之后,大家就可以在易语言中对 API 函数进行调用,进一步完成程序所需的操作了。

内部 API 一般指操作系统自带的 DLL 中的 API 函数,随操作系统安装就可以调用的,如 Windows 自带的 DLL 文件。而外部 API 一般指公司或个人提供的 DLL 文件,需要另外安装到相应的目录下。下面先学习内部 API。

### 13.3.1 内部 API

例程一:

调用 API 函数,可以实现很多特殊的效果,下面是使用 3 个 DLL 命令实现窗体淡入淡出效果的实例。

1. 打开易语言,新建一个易程序。如图 13-4 所示。

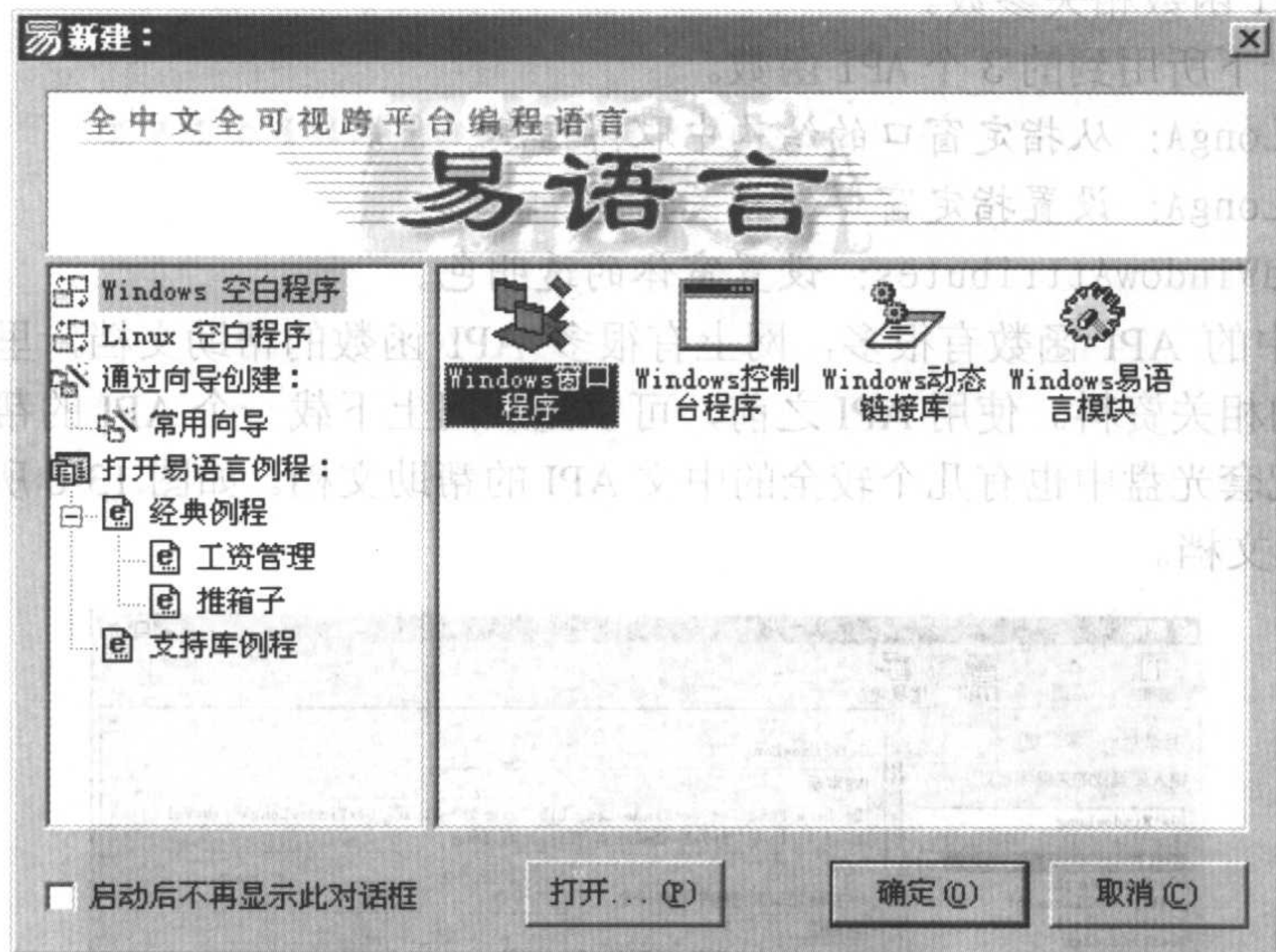


图 13-4 新建一个易程序

2. 在“\_启动窗口”中添加一个滑块条。如图 13-5 所示。

设置滑块条相应的属性值:

方向: 横向

刻度类型: 下/右

单位刻度值: 1

最小位置: 0

最大位置: 255

位置: 255



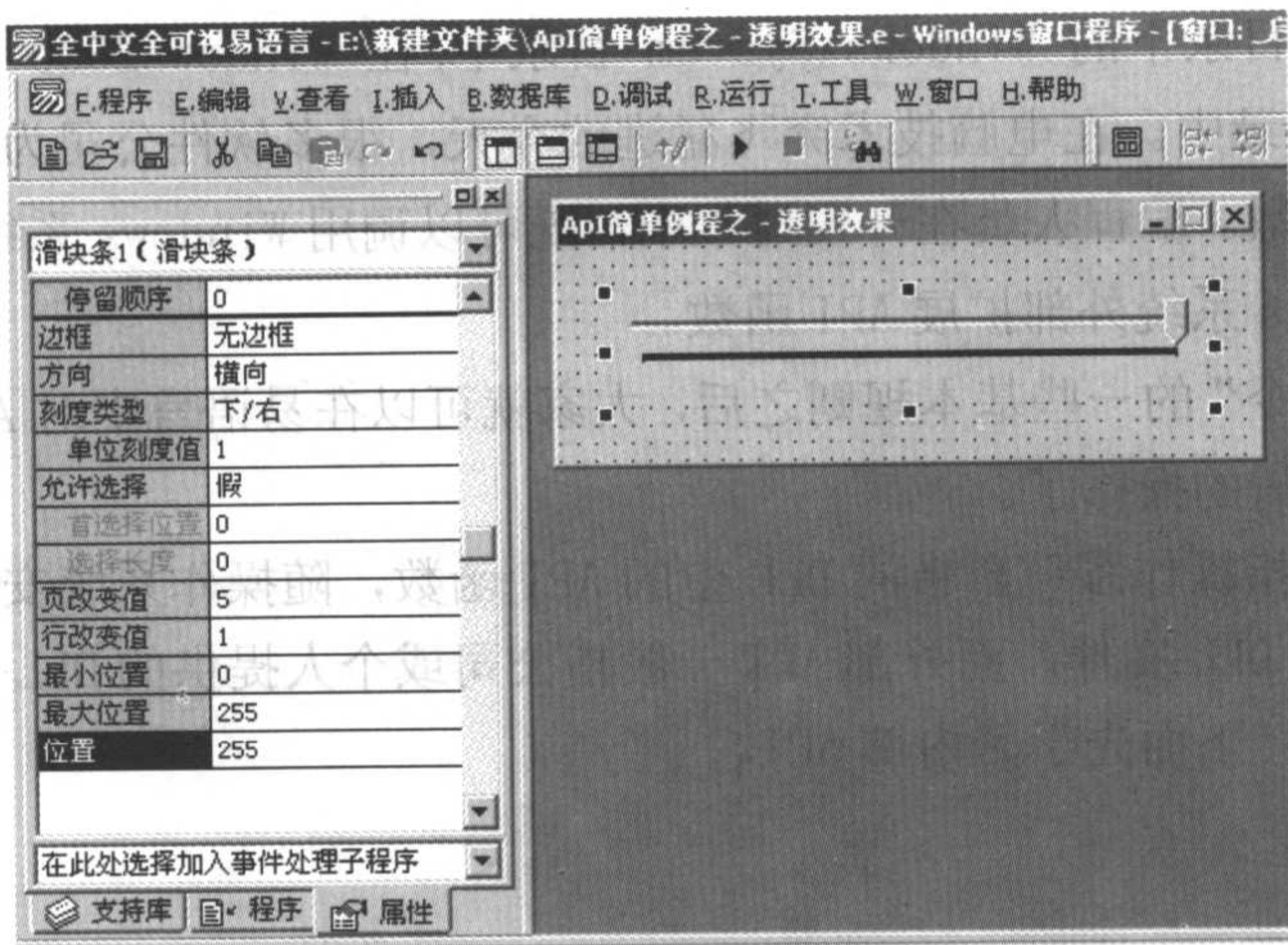


图 13-5 添加滑块条并进行相应属性设置

3. 双击“程序”面板中的“DLL 命令”选项，在代码设计工作区中单击鼠标右键，在弹出菜单中选择“新 DLL 命令”选择项，新建 DLL 命令，用于调用 API 函数。

4. 添加 API 函数相关参数。

先来了解一下所用到的 3 个 API 函数。

GetWindowLongA: 从指定窗口的结构中取得信息。

SetWindowLongA: 设置指定窗口的相关信息。

SetLayeredWindowAttributes: 设置窗体的透明色。

Windows 中的 API 函数有很多，网上有很多 API 函数的帮助文档，里面列举了常用 API 函数指令的相关资料。使用 API 之前，可以先从网上下载一个 API 的帮助文档以备查询之用。本书配套光盘中也有几个较全的中文 API 的帮助文档。如图 13-6 所示，就是一个 API 函数的帮助文档。

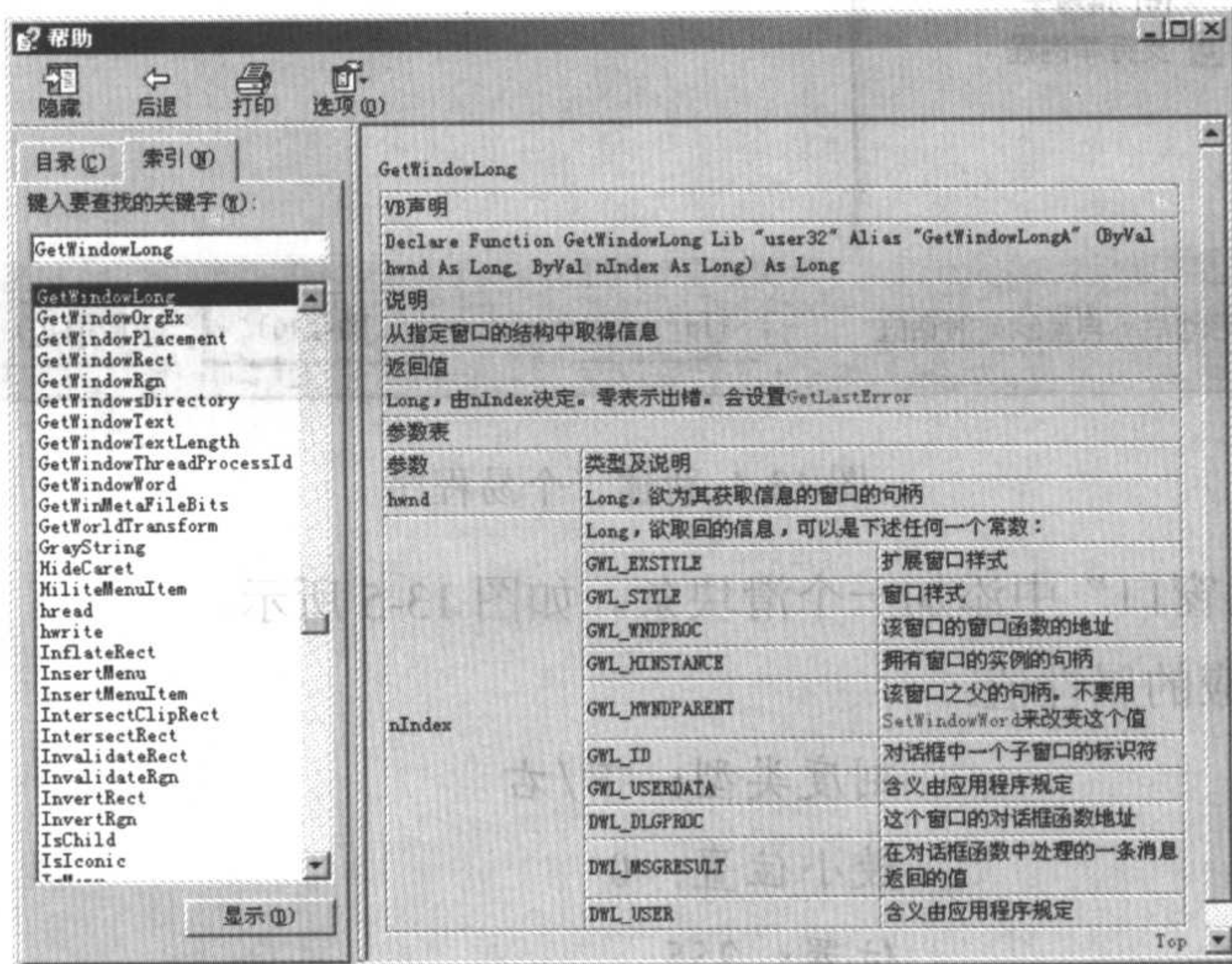


图 13-6 API 函数帮助文档



以“GetWindowLongA”为例，其 VB 函数声明为：

```
Declare Function GetWindowLong Lib "user32" Alias "GetWindowLongA" (ByVal hwnd As Long, ByVal nIndex As Long) As Long
```

“GetWindowLong”是函数的名称；“user32”为这个函数命令所在的 DLL 库文件名称（不可被修改，如果被修改，DLL 命令将无法找到所指定的 DLL 库）；“GetWindowLongA”是在 DLL 库中对应的命令名。“ByVal hwnd As Long, ByVal nIndex As Long”里“hwnd”和“nIndex”所指的是 DLL 所使用的参数名称（可被修改，参数个数、位置与类型要一致）；其中“ByVal”代表以“传值”（区别于“传址”）方式传递参数；每个参数后的“As long”指的是参数数据类型；最后的“As Long”是 DLL 命令返回值的数据类型。

易语言中的“整数型”数据类型与 VB 中的“long”数据类型相对应。

添加 3 个 DLL 命令如下：

Dll命令名	返回值类型	备 注		
GetWindowLong	整数值			
Dll库文件名				
在Dll库中对应命令名				
GetWindowLongA				
参数名	类 型	传址	数组	备 注
hwnd	整数值			
nIndex	整数值			

Dll命令名	返回值类型	备 注		
SetWindowLong	整数值			
Dll库文件名				
在Dll库中对应命令名				
SetWindowLongA				
参数名	类 型	传址	数组	备 注
hwnd	整数值			
nIndex	整数值			
dwNewLong	整数值			

Dll命令名	返回值类型	备 注		
SetLayeredWindowAttributes	整数型			
Dll库文件名				
在Dll库中对应命令名				
SetLayeredWindowAttributes				
参数名	类 型	传址	数组	备 注
hwnd	整数型			
crKey	整数型			
bAlpha	整数型			
dwFlags	整数型			





5. 回到“\_启动窗口”，双击窗体自动建立“\_启动窗口\_创建完毕”子程序，然后新建一个整数型的子程序变量“Ret”。

在子程序中输入代码：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

变量名	类型	静态	数组	备注
Ret	整数型			

```
Ret = GetWindowLong (取窗口句柄 0, -20)
Ret = 位或 (Ret, 524288)
SetWindowLong (取窗口句柄 0, -20, Ret)
SetLayeredWindowAttributes (_启动窗口.取窗口句柄 0, 0, 255, #LWA_ALPHA)
```

6. 回到“\_启动窗口”，选择“滑块条”在滑块条的事件列表中选择“位置被改变”事件，产生“\_滑块条1\_位置被改变”子程序，在其中输入代码：

子程序名	返回值类型	公开	备注
滑块条1_位置被改变			

```
SetLayeredWindowAttributes (_启动窗口.取窗口句柄 0, 0, 滑块条1.位置, #LWA_ALPHA)
```

7. 按下“F5”键运行程序，然后拖动滑块条，可以看到，窗体渐渐变得透明，直到完全透明。如图 13-7 所示。

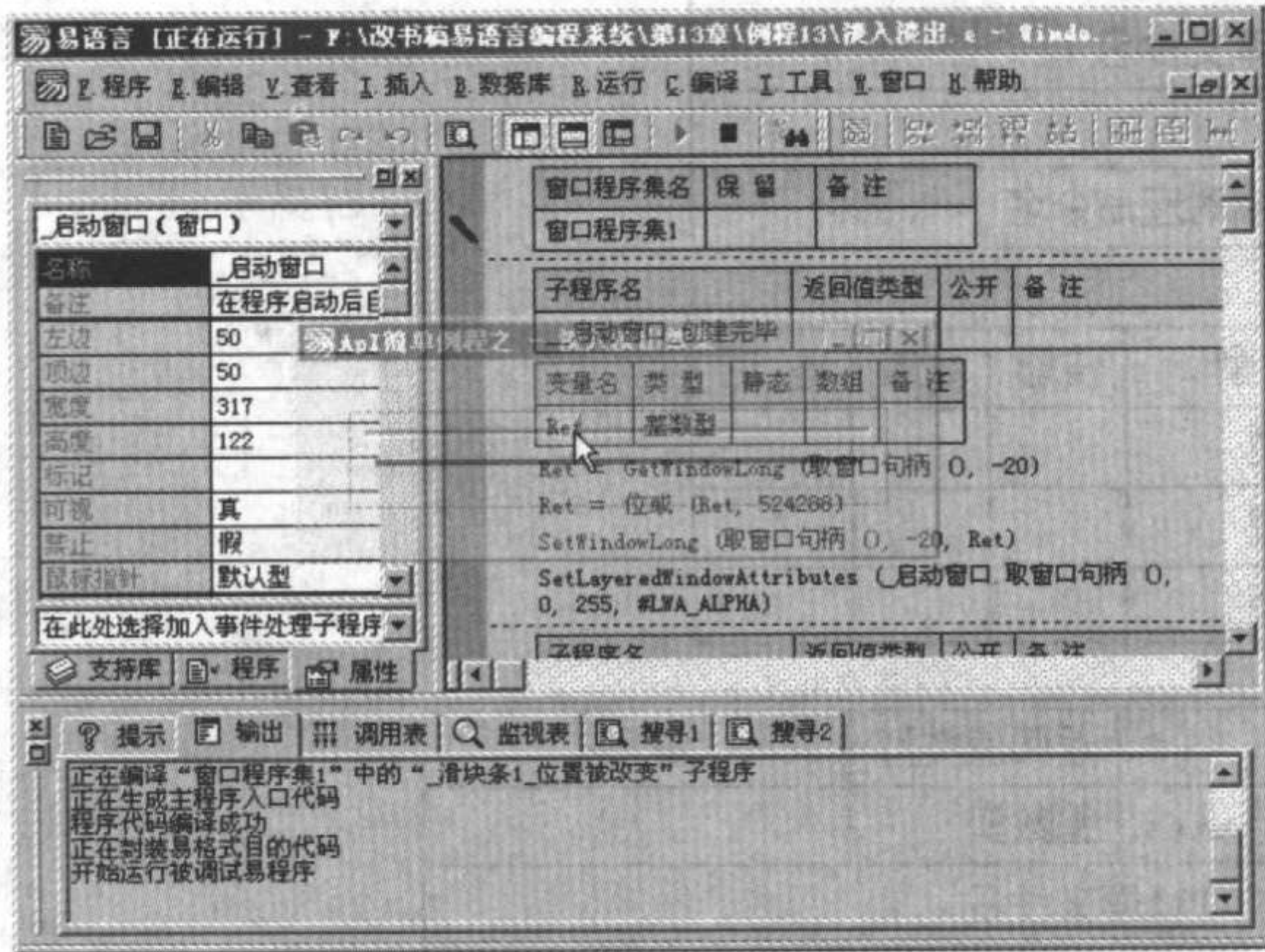


图 13-7 运行后窗口淡入淡出的效果

上述程序 DLL 表中的“DLL 命令名”也可以用中文名称代替原有的英文名称。

### 例程二：

大家再看一个可以在三秒钟后关闭窗口的例程。在这个例程中有两个可执行文件，一个名为 play.exe, 另一个名为 stop.exe。通过运行 play.exe 程序后将主窗口的句柄发送给



所调用的 stop.exe 程序，stop.exe 程序三秒钟后根据句柄关闭 play.exe 程序的主窗口。

1. 新建一个易程序，双击“\_启动窗口”界面，进入“\_\_启动窗口\_创建完毕”子程序，在其中输入代码：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

运行 (“STOP.EXE ” + 到文本 (启动窗口.取窗口句柄 ()), 假, )

注意：上述代码中，STOP.EXE 后紧跟一个空格。

2. 选择“编译”菜单下的“编译”项编译程序，编译文件名为“play.exe”。

3. 再新建一个易程序，在“\_启动窗口”中添加一个时钟组件。如图 13-8 所示。



图 13-8 在“\_启动窗口”中添加一个时钟组件

4. 新建两个 DLL 命令，添写相应的 API 函数信息。

在本例程的 stop.exe 中用到了两个 API 函数，分别是 IsWindow 和 SendMessage。（对于函数的详细说明，请根据上文所提到查找 API 函数帮助的方法自行查看）在此程序中使用了一个时钟组件，通过时钟组件的“时钟周期”事件来控制关闭窗口的时间间隔。

两个 API 函数的说明：

IsWindow：判断一个窗口句柄是否有效。返回值类型为 Long，如果成功返回非零数，失败返回零。

SendMessage：向系统发送消息函数。除非消息处理完毕，否则该函数不会返回。返回值类型为 Long，返回数值的多少由具体的消息决定。

Dll命令名	返回值类型	备 注		
送消息	整数型	对应SendMessageA		
Dll库文件名				
在Dll库中对应命令名				
SendMessageA				
参数名	类 型	传址	数组	备 注
hwnd	整数型			
wmsg	整数型			
wparam	整数型			
lparam	整数型			





Dll命令名	返回值类型	备 注		
窗口句柄是否有效	整数型	对应IsWindow		
Dll库文件名				
在Dll库中对应命令名				
IsWindow				
参数名	类 型	传址	数组	备 注
hwnd	整数型			

5. 双击“\_启动窗口”窗体，进入代码设计工作区。在窗口程序集中新建一个文本型程序集变量“命令行”，并在“\_\_启动窗口\_创建完毕”子程序中输入如下代码。

窗口程序集名	保留	备注	
窗口程序集1			
变量名	类型	数组	备注
命令行	文本型	1	

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

取命令行 (命令行)

如果真 (取数组成员数 (命令行) > 0)

时钟1.时钟周期 = 3 × 1000

3秒钟后关闭窗口，结束程序

置托盘图标 (#小图标, “自动关闭窗口”)

6. 回到“\_启动窗口”界面，双击时钟组件，在“\_时钟 1\_周期事件”子程序中输入如下代码：

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

时钟1.时钟周期 = 0

如果真 (窗口句柄是否有效 (到数值 (命令行 [1])) > 0)

是窗口句柄

送消息 (到数值 (命令行 [1]), 16, 0, 0)

置托盘图标 ({ }, )

结束 0

7. 选择“编译”菜单下的“编译”项编译程序，编译文件名为“stop.exe”。

8. 运行编译好的 play.exe 程序，可以看到 3 秒钟后窗口被 stop.exe 程序自动关闭。



程序 play.exe 调用程序 stop.exe，且传送 play.exe 的主窗口句柄给程序 stop.exe，程序 stop.exe 接收到程序 play.exe 的主窗口句柄，使用时钟组件三秒后 API 函数关闭了程序 play.exe 的主窗口，并且也结束了程序 stop.exe 的运行。

### 13.3.2 外部 API

下面是调用外部 DLL 库中的 API 函数，来实现一个鼠标滑动产生水波的特效。

该例程要使用到一个外部的 DLL 库，所以，在编写程序前，请先从随书光盘中将 WaterDll.dll 和例图 01.bmp 文件拷贝到易语言程序所在目录下。

本例程要用到系统 DLL 库 user32.dll 中的 GetDC 和 LoadImage 命令函数，和 3 个外部 DLL 库中的命令。

1. 新建一个易程序，在“\_启动窗口”中添加一个时钟组件。如图 13-9 所示。

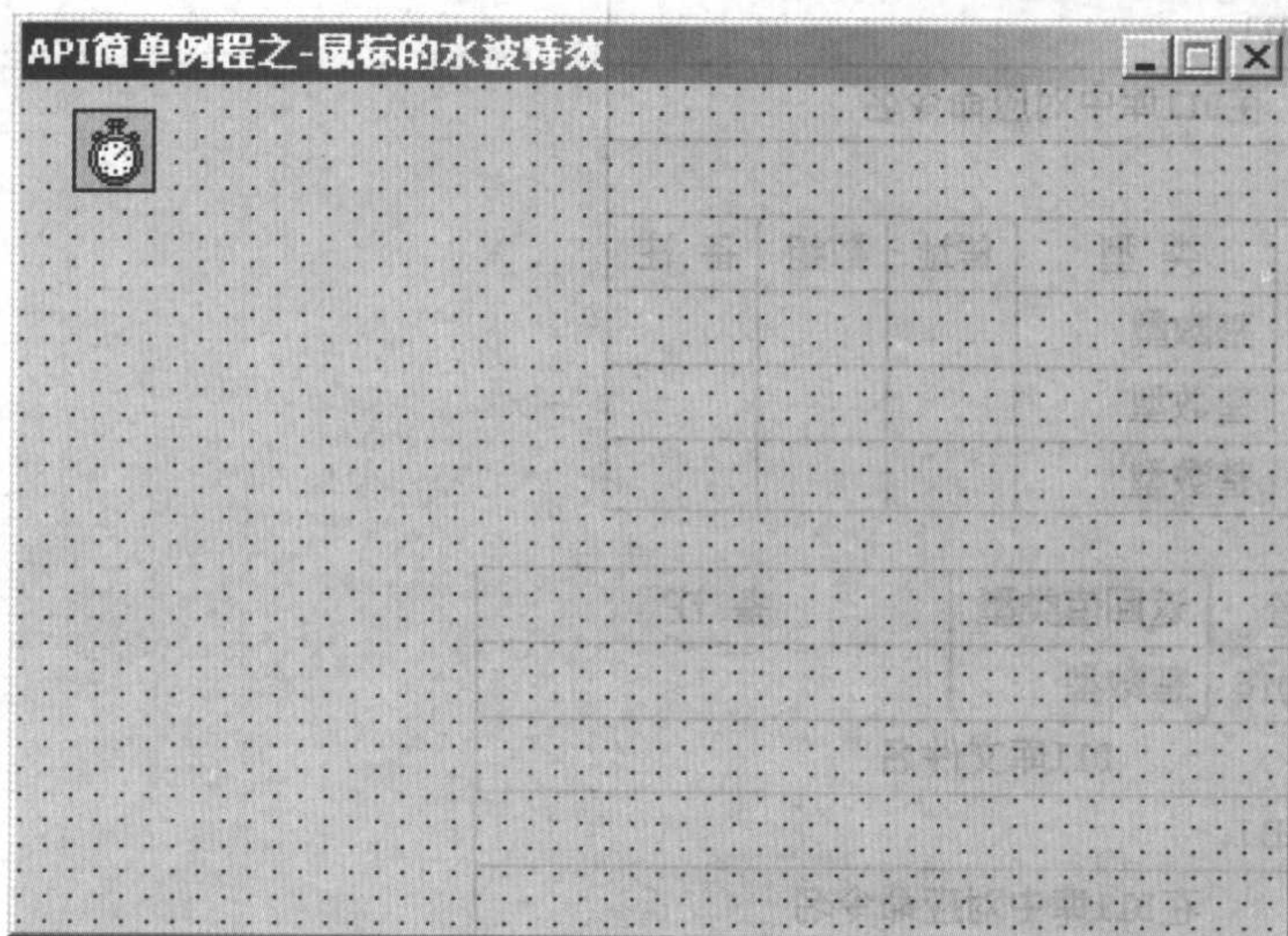


图 13-9 在“\_启动窗口”中添加时钟组件

2. 添加 API 函数相关参数。

Dll命令名	返回值类型	备 注		
初始化水纹	整数型			
Dll库文件名				
waterdll.dll				
在Dll库中对应命令名				
WaterInit				
参数名	类 型	传址	数组	备 注
位图句柄	整数型			





Dll命令名	返回值类型	备 注		
取设备场景	整数型			
Dll库文件名				
在Dll库中对应命令名				
GetDC				
参数名	类 型	传址	数组	备 注
窗口句柄	整数型			

Dll命令名	返回值类型	备 注		
水纹时间	整数型			
Dll库文件名				
waterdll.dll				
在Dll库中对应命令名				
WaterTimer				
参数名	类 型	传址	数组	备 注
设备场景	整数型			
源左边	整数型			
源右边	整数型			

Dll命令名	返回值类型	备 注		
水纹鼠标动作	整数型			
Dll库文件名				
waterdll.dll				
在Dll库中对应命令名				
WaterMouseAction				
参数名	类 型	传址	数组	备 注
设备场景	整数型			
源左边	整数型			
源右边	整数型			
目标左边	整数型			
目标右边	整数型			
半径	整数型			扩展半径
深度	整数型			扩展深度

3. 双击启动窗口窗体, 自动生成“\_\_启动窗口\_创建完毕”子程序, 在“\_\_启动窗口\_创建完毕”子程序中输入如下代码:



子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
位图句柄	整数型			
位图对象	对象			

时钟1.时钟周期 = 10

位图对象.创建图片对象 (\_\_启动窗口.底图)

位图句柄 = 位图对象.读数值属性 ("Handle", )

初始化水纹 (位图句柄)

4. 双击时钟组件，在“\_时钟1\_周期事件”子程序中输入如下代码：

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

水纹时间 (取设备场景 (\_\_启动窗口.取窗口句柄 0), 图片左边, 图片顶边)

5. 在启动窗口的事件选择框中选择“鼠标位置被移动”选项，在“\_\_启动窗口\_鼠标位置被移动”子程序中输入如下代码：

子程序名	返回值类型	公开	备 注		
__启动窗口_鼠标位置被移动	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

水纹鼠标动作 (取设备场景 (\_\_启动窗口.取窗口句柄 0), 图片左边, 图片顶边, 横向位置, 纵向位置, 5, 80)

在启动窗口的事件选择框中选择“鼠标左键被按下”选项，在“\_\_启动窗口\_鼠标左键被按下”子程序中输入如下代码：

子程序名	返回值类型	公开	备 注		
__启动窗口_鼠标左键被按下	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

水纹鼠标动作 (取设备场景 (\_\_启动窗口.取窗口句柄 0), 图片左边, 图片顶边, 横向位置, 纵向位置, 60, 500)





6. 运行程序，在窗体上移动鼠标，就可以看到逼真的水波效果了。如图 13-10 所示。



图 13-10 鼠标滑过窗体所产生的水波效果

## 13.4 本章小结

API 函数由操作系统提供，可以实现一般组件所不能实现的功能。

大家学习本章主要的目的是先看懂 API 手册中的说明，只有理解了这些 API 说明，才能更好地使用它们，这些 API 手册可在网上搜索下载。其次是正确使用 API，将英文的 API 函数命令汉化后使用，做到全中文编程。

易语言也可以编写标准动态链接库（DLL 文件），这些将在专门的章节中再详细介绍。大家还可以进行以下的练习：

1. 试学习编写本章的水波特效例程。
2. 找一份比较全面的中文 API 函数帮助文档，并学习在易语言中使用。
3. 使用 API 函数，制作模拟键盘按下“Win 键+D”，和模拟鼠标移动的程序。

主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称
主 窗 体	子 窗 体	窗 体 类 型	窗 体 名 称



## 第十四章 易模块

### 14.1 易模块的作用

在大型软件项目的开发过程中，需要分工协作进行系统性开发。易语言提供了模块化开发支持——“易模块”。通过使用易模块，编程人员可以将常用的代码封装起来并重复使用到其他程序中，或提供给第三方用于开发使用，或用作开发大型软件项目中的某一部分，然后在软件项目的封装阶段将所有这些模块组织编译成一个完整程序。

### 14.2 易模块的调用方法

对于已经编译好的易模块，要正确安装到系统中后才可以被其他程序调用。下面通过一个简单的例程来了解易模块的使用方法。

这个例程是通过调用一个易模块中的子程序来打开“浏览文件夹”对话框，并把所选择文件夹的路径显示在编辑框中。

1. 从“工具”菜单中选择“易模块管理”。如图 14-1 所示。

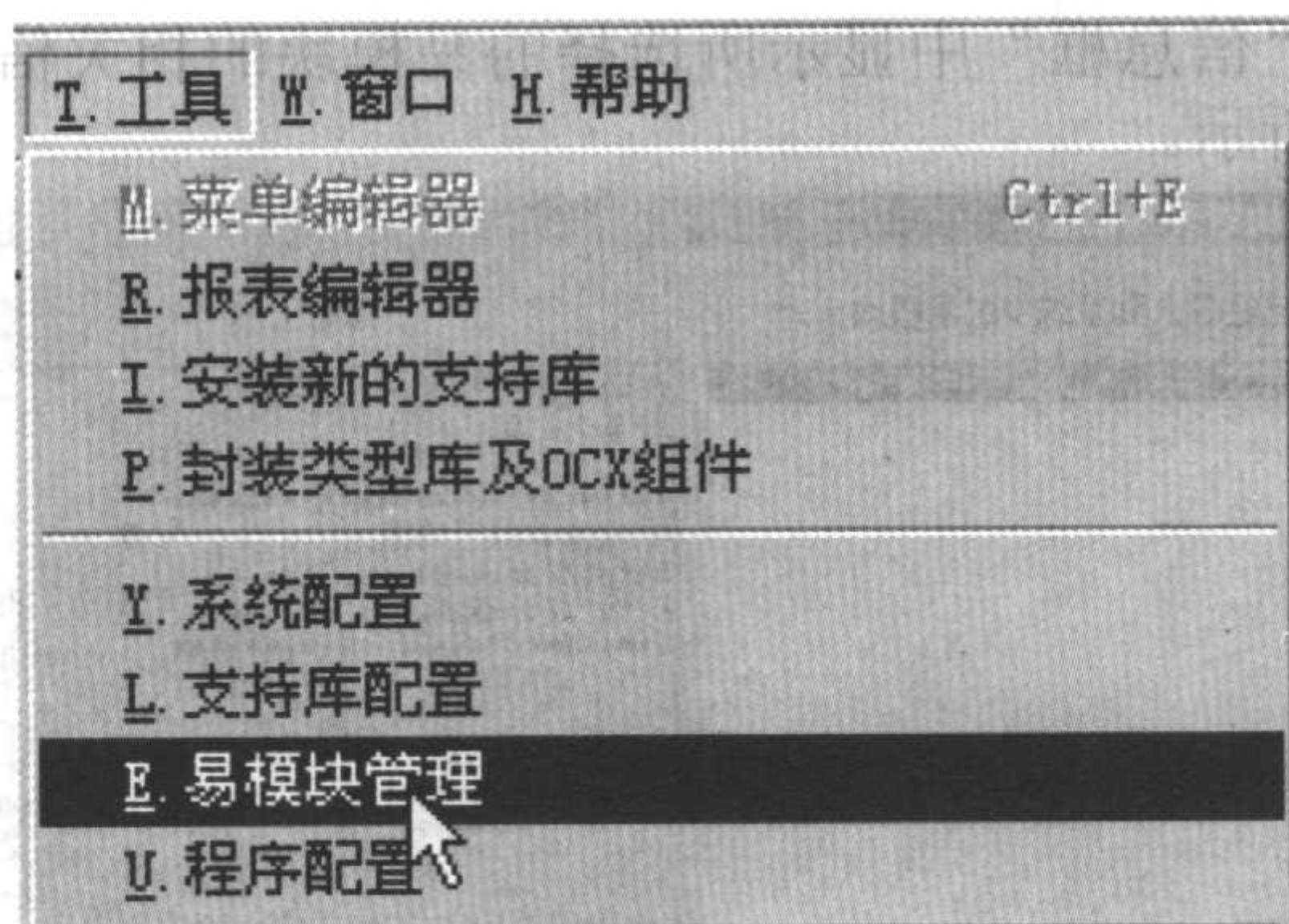


图 14-1 选择“工具”菜单下的“易模块管理”

2. 在弹出的“易模块管理对话框”中，单击“导入新模块”。如图 14-2 所示。



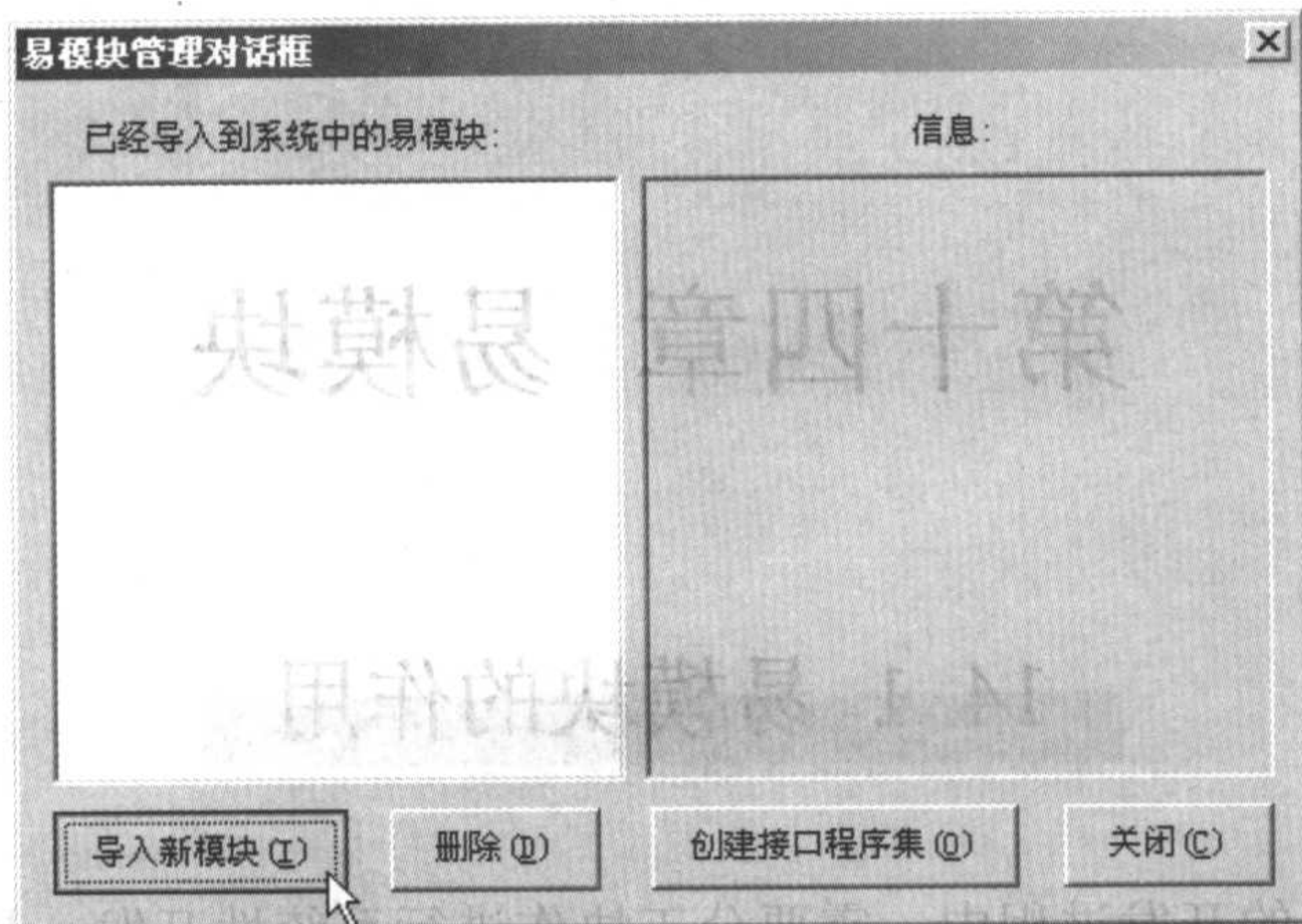


图 14-2 易模块管理对话框

3. 在弹出的易模块选择对话框中，选择“浏览文件夹.ec”文件（易模块文件），此文件可在随书光盘的本章目录中找到。如图 14-3 所示。

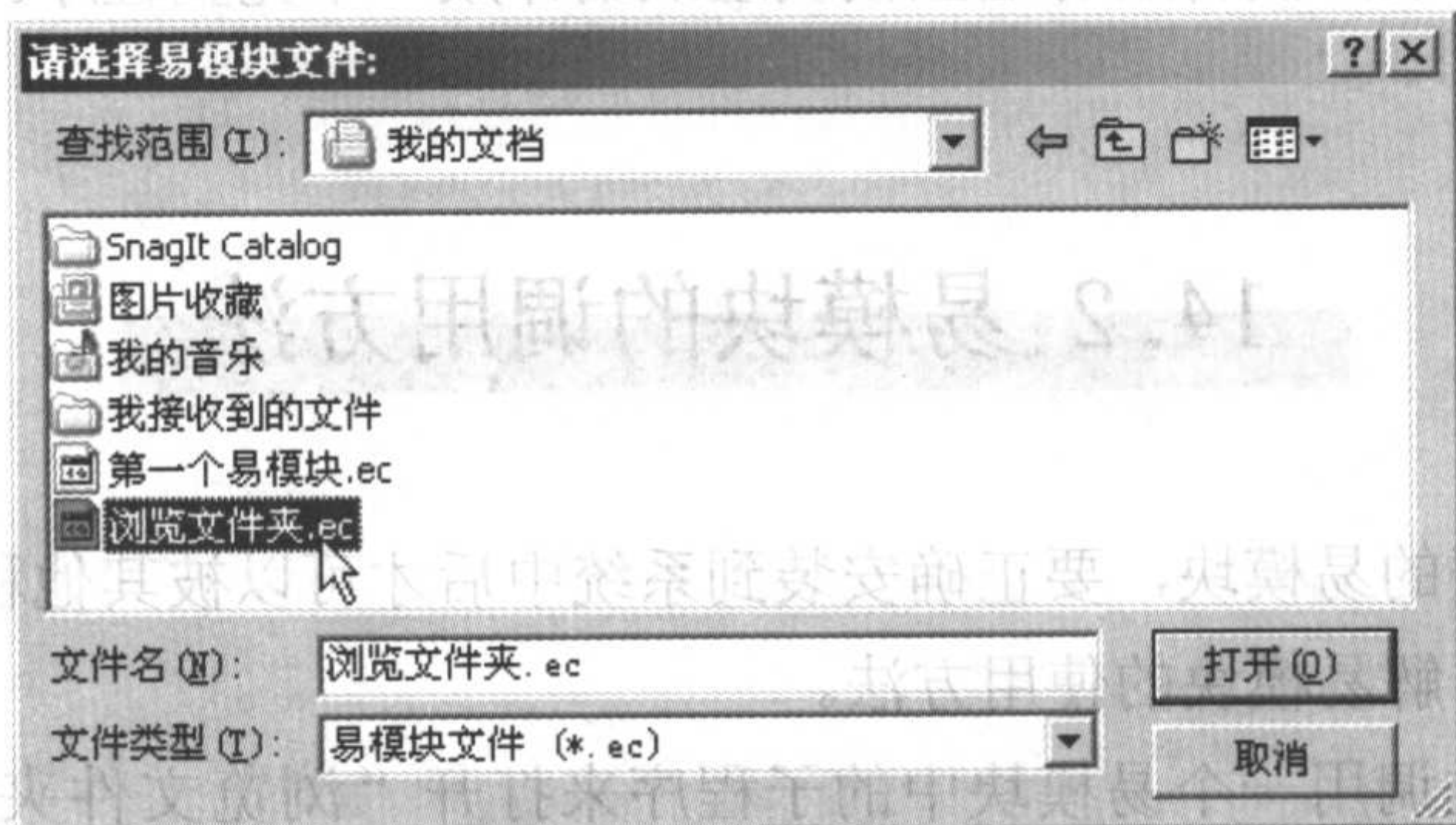


图 14-3 易模块选择对话框

4. 此模块被导入到“易模块管理对话框”中，左边的“易模块”列表中显示已被导入的易模块名称，右边的“信息框”中显示所选择的易模块的相关信息。然后点击“创建接口程序集”。如图 14-4 所示。

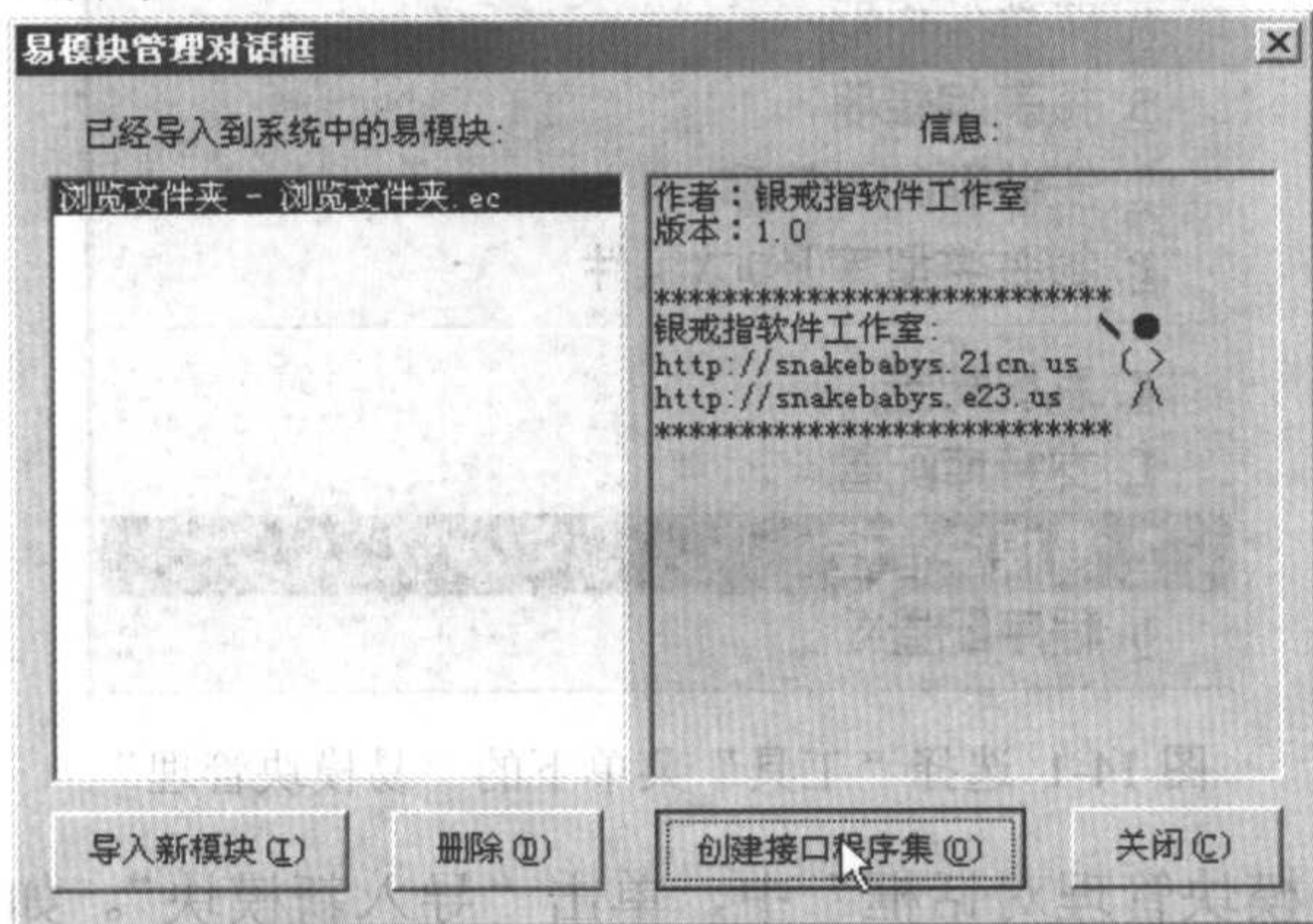


图 14-4 导入易模块之后的“易模块管理对话框”



5. 在弹出的“创建接口程序集”中，左边的“接口程序”列表中显示的是该模块的外部接口子程序名，右边的“信息”框中显示的是所选接口子程序的调用方法和相关说明。如图 14-5 所示。

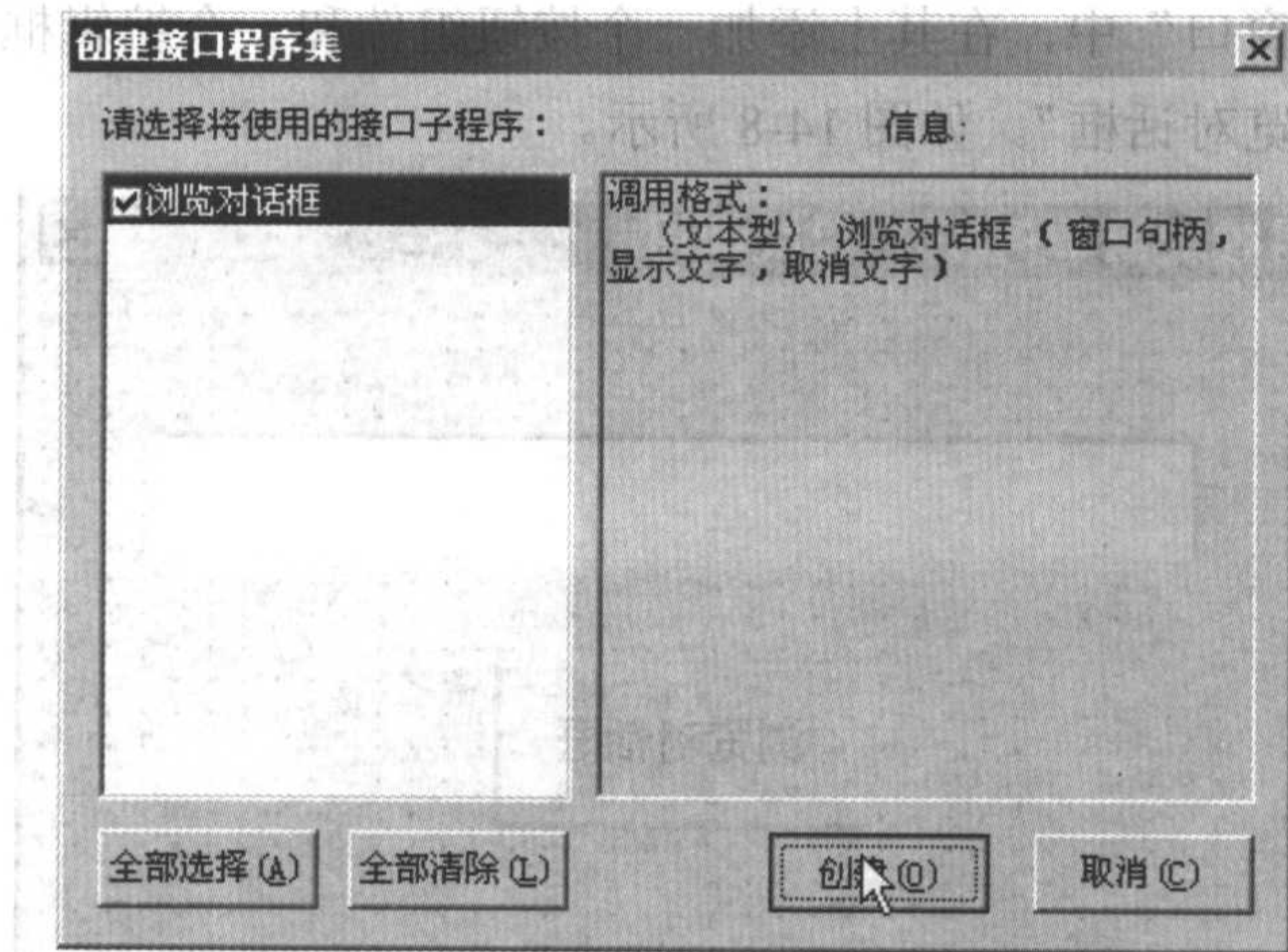


图 14-5 “创建接口程序集”对话框

6. 点击“创建”。易语言的代码编辑区中会自动生成接口程序集。模块的程序集名称，也就是模块的文件名称，前面被自动加上“\_模块\_”标记，以提示此程序集是模块。如图 14-6 所示。

程序集名		备注			
_模块_浏览文件夹		** 不要更改此处 浏览文件夹.ec			

子程序名	返回值类型	公开	备注		
浏览对话框	文本型				
参数名	类型	参考	可空	数组	备注
窗口句柄	整数型	✓			所在窗口句柄，注意不是控件句柄
显示文字	文本型	✓			
取消文字	文本型	✓			放弃选择时的文字

‘本子程序功能由系统自动转交对应模块实现，可以删除但不能修改。

图 14-6 代码编辑区中接口程序集

同时在程序管理面板中也会自动生成一个“模块程序集”。如图 14-7 所示。

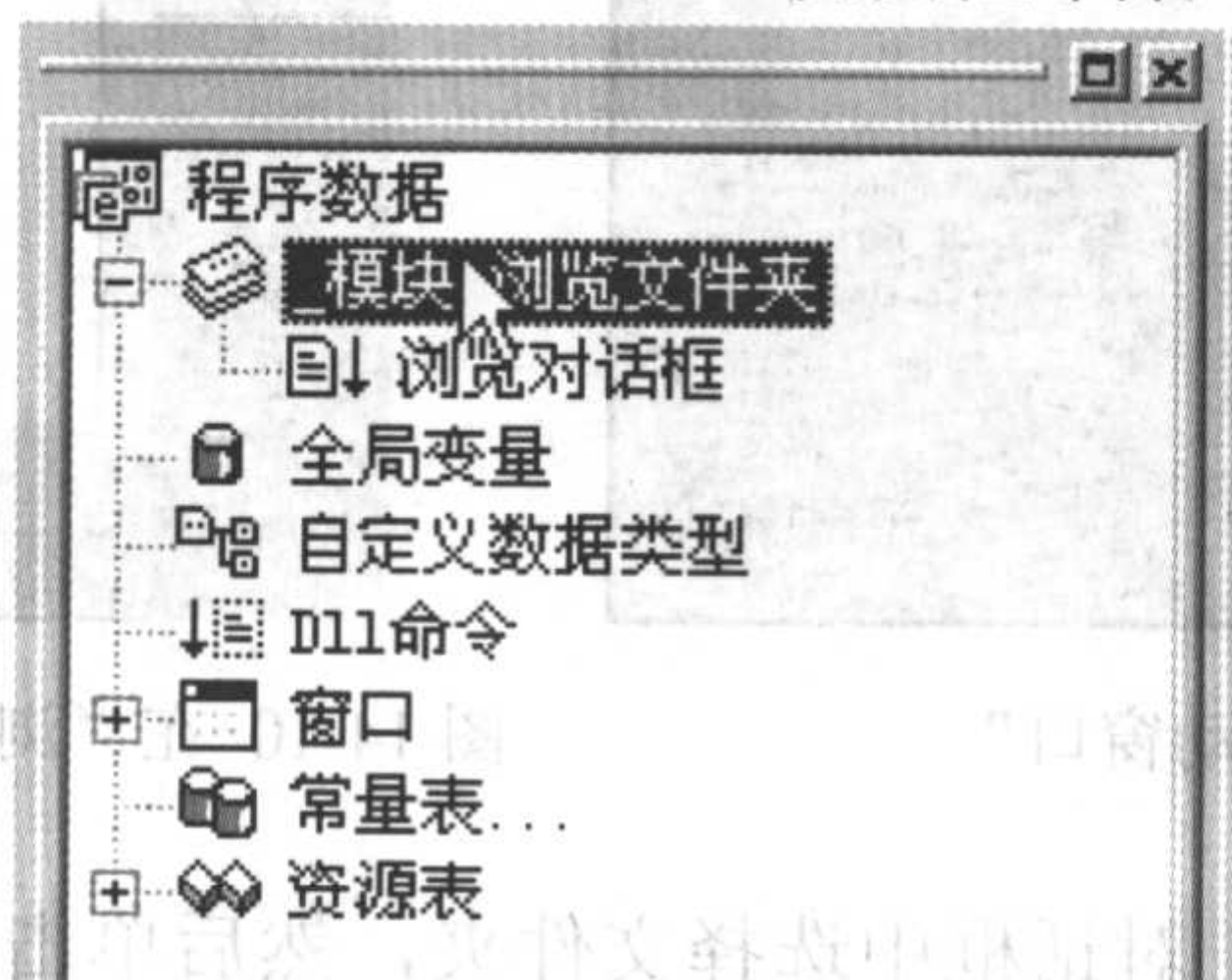


图 14-7 自动创建模块接口程序集





**注意：**模块程序集创建完毕后，可以直接使用该程序集中的子程序。在编译易程序时，所有被使用的易模块会一同被编译。

7. 回到“\_启动窗口”中，在其上添加一个按钮组件和一个编辑框组件，设置按钮组件的标题属性为“浏览对话框”。如图 14-8 所示。

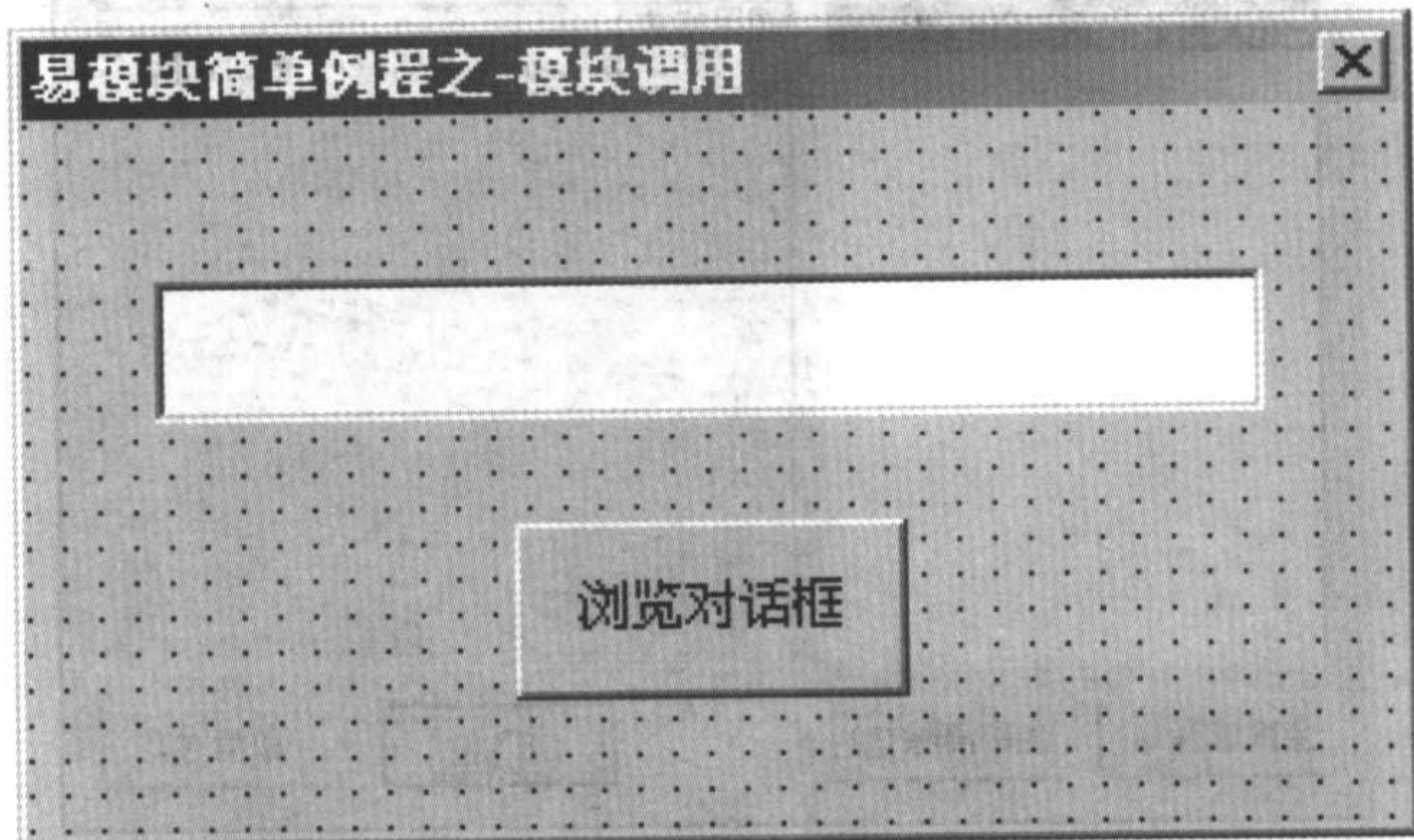


图 14-8 在“\_启动窗口”添加的组件

8. 双击“浏览对话框”按钮，在“\_按钮 1\_被单击”事件子程序中调用模块中所提供的子程序“浏览对话框()”，并添写其中的参数：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

编辑框1.内容 = 浏览对话框 (启动窗口.取窗口句柄 0, “文件夹游览”, “”)

9. 按下“F5”键，试运行。在“\_启动窗口”中按下“浏览对话框”按钮。如图 14-9 所示。

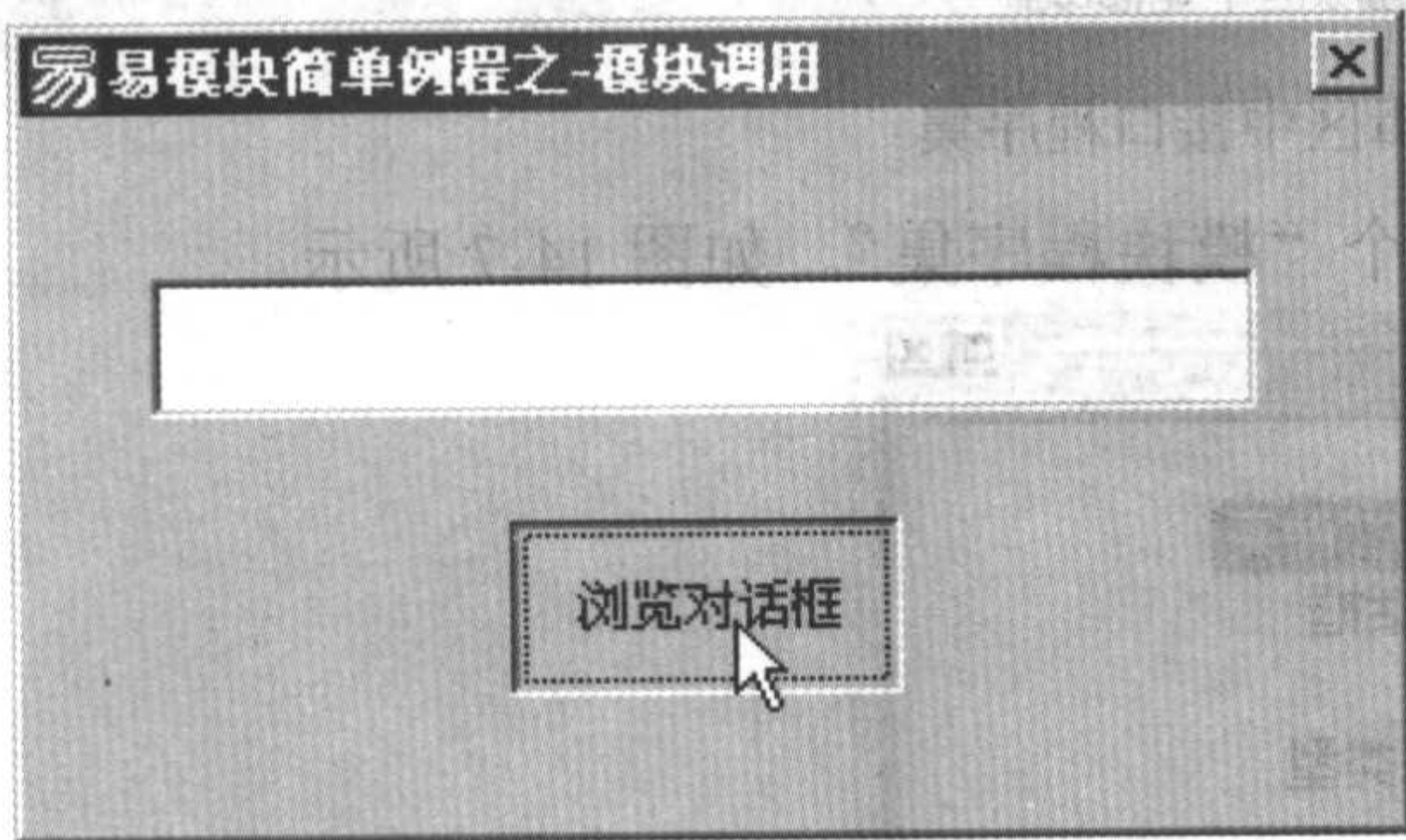


图 14-9 运行后的“\_启动窗口”

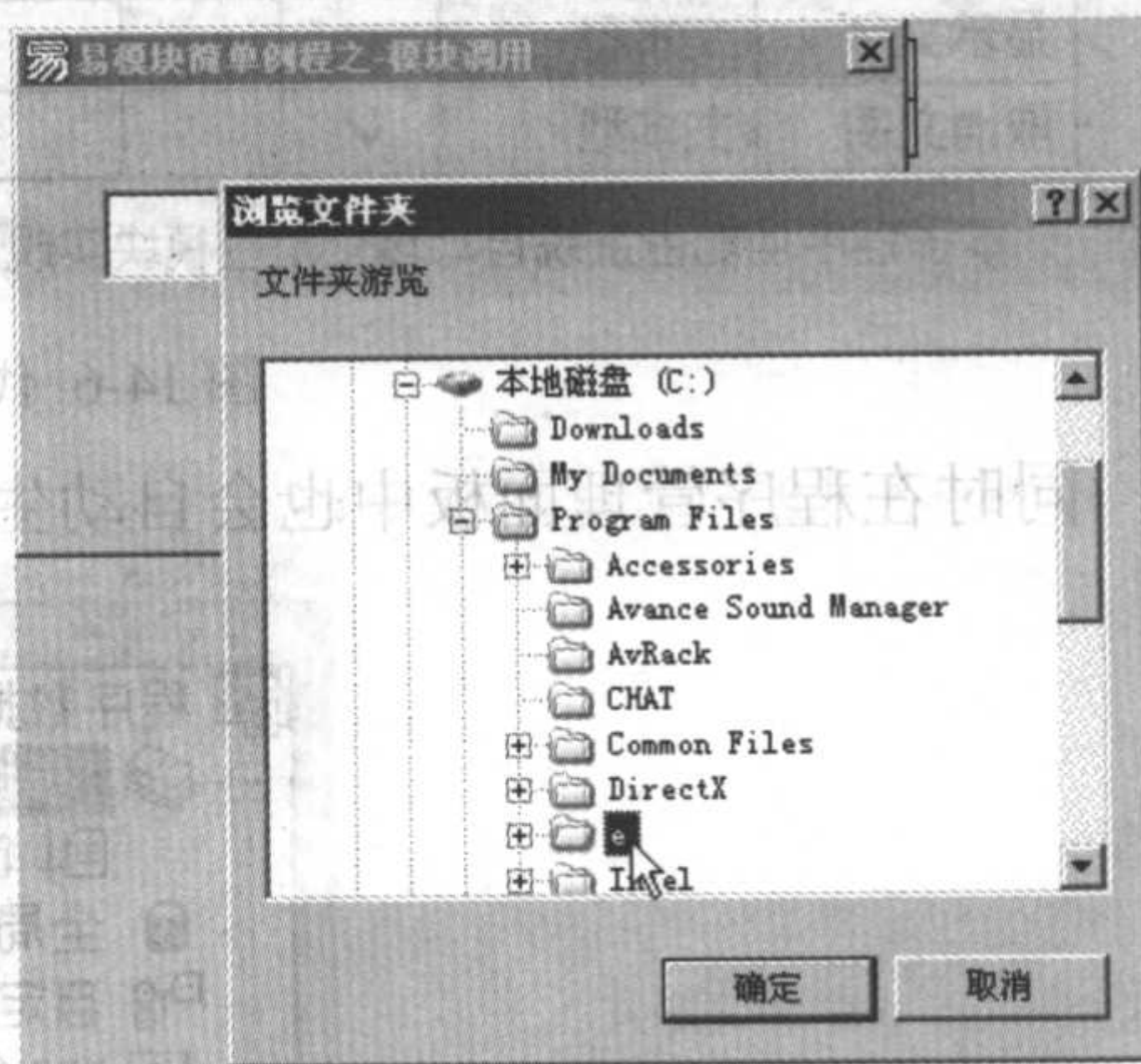


图 14-10 在“浏览文件夹”对话框中选择文件夹

在弹出的“浏览文件夹”对话框中选择文件夹，然后单击“确定”。如图 14-10 所示。

在编辑框中就会显示出所选择的文件夹的路径。如图 14-11 所示。



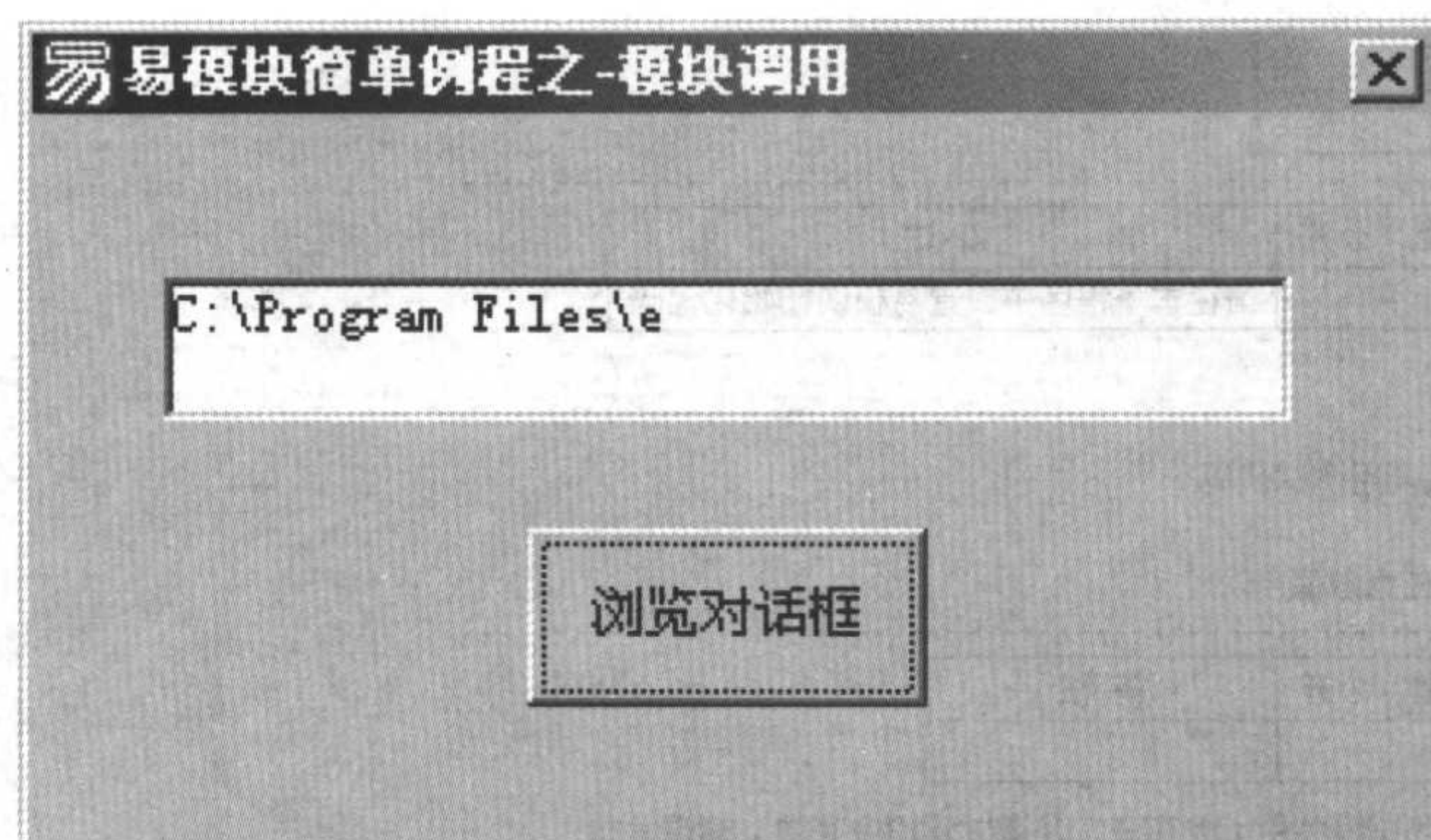


图 14-11 在编辑框中显示所选文件夹的路径

## 14.3 易模块的开发与编译

使用者可以自行创建自己所需要的易模块，然后在易语言中导入使用。下面就来了解一下易模块的创建方法。

### 14.3.1 易模块的开发

1. 从“程序”菜单中选择“新建”，在“新建”窗口中选择“Windows 易语言模块”，然后点击“确定”按钮。如图 14-12 所示。

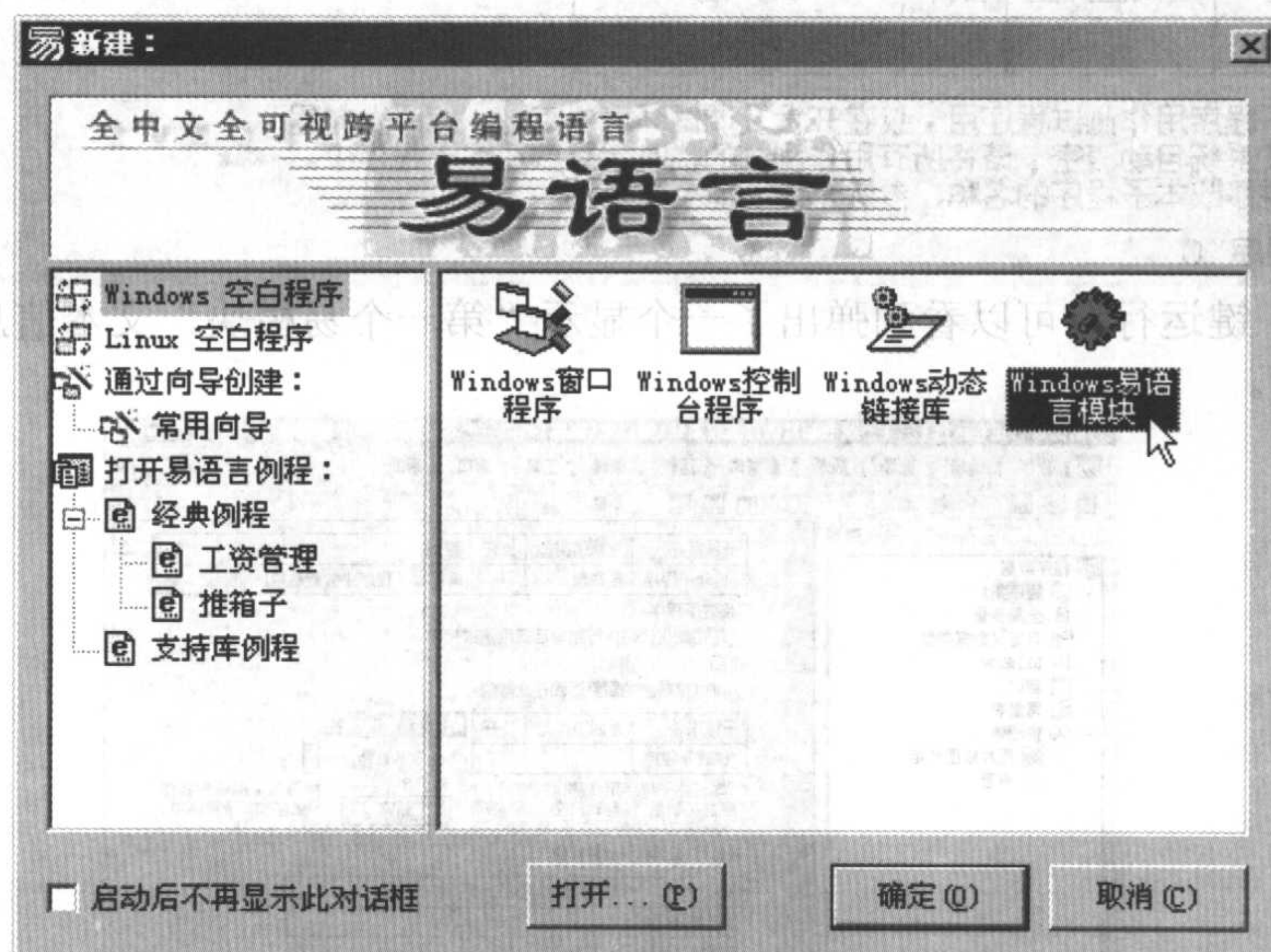


图 14-12 新建一个 Windows 易语言模块

进入代码编辑区，易语言会自动创建两个子程序，如图 14-13 所示。



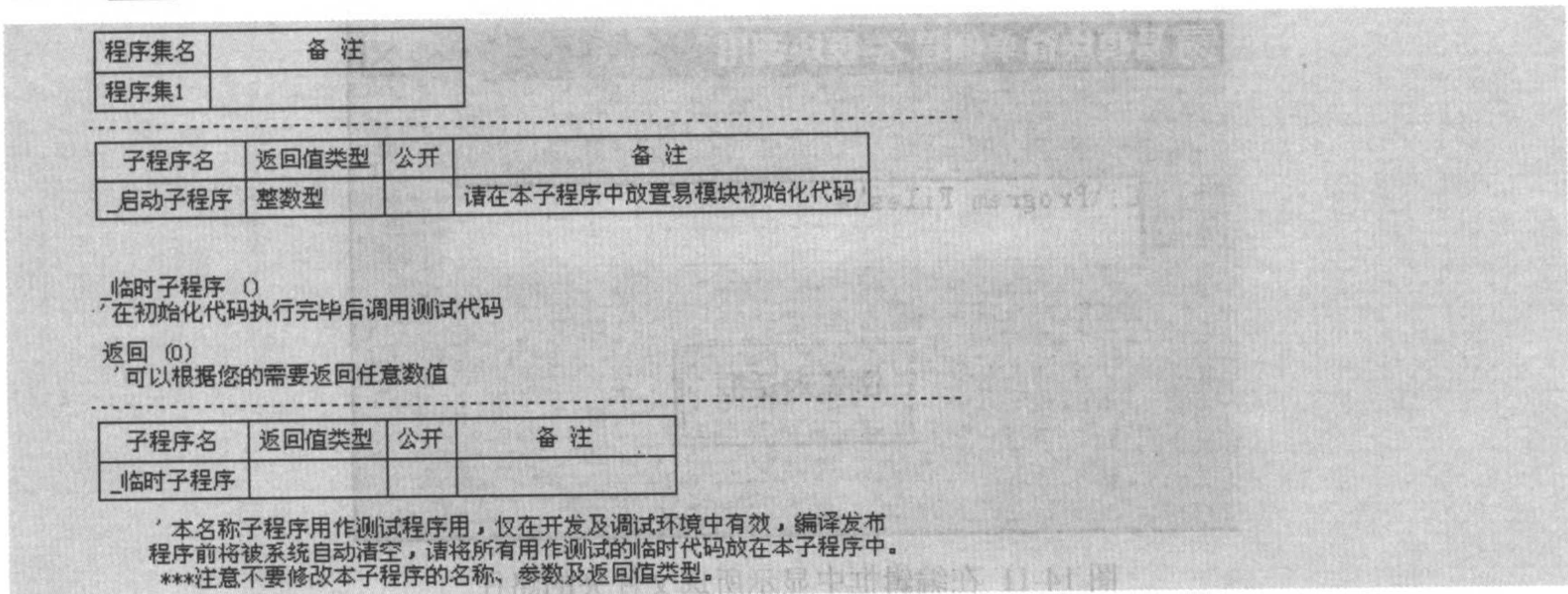


图 14-13 易模块开发的相应子程序

“\_启动子程序”：此子程序负责进行易模块的初始化。

“\_临时子程序”：用做调试易模块之用。所有在该子程序中书写的程序代码，仅在调试环境下运行有效。在编译易模块时，该子程序中的所有代码将不会被编译到模块文件中。

2. 在此程序集中新建一个子程序，改名为“信息框子程序”，并将公开属性打上对勾。输入以下代码：

子程序名	返回值类型	公开	备注
信息框子程序		<input checked="" type="checkbox"/>	

信息框 (“第一个易模块”, 0, )

注意：上述代码中，一定要将公开属性打上对勾，以显示是公开的接口程序。

在“\_临时子程序”中，输入以下代码：

子程序名	返回值类型	公开	备注
_临时子程序			

本名称子程序用作测试程序用，仅在开发及调试环境中有效，编译发布程序前将被系统自动清空，请将所有用作测试的临时代码放在本子程序中。 \*\*\*  
注意不要修改本子程序的名称、参数及返回值类型。

信息框子程序 0

3. 按 F5 键运行，可以看到弹出了一个显示“第一个易模块”文本信息的信息框。如图 14-14 所示。

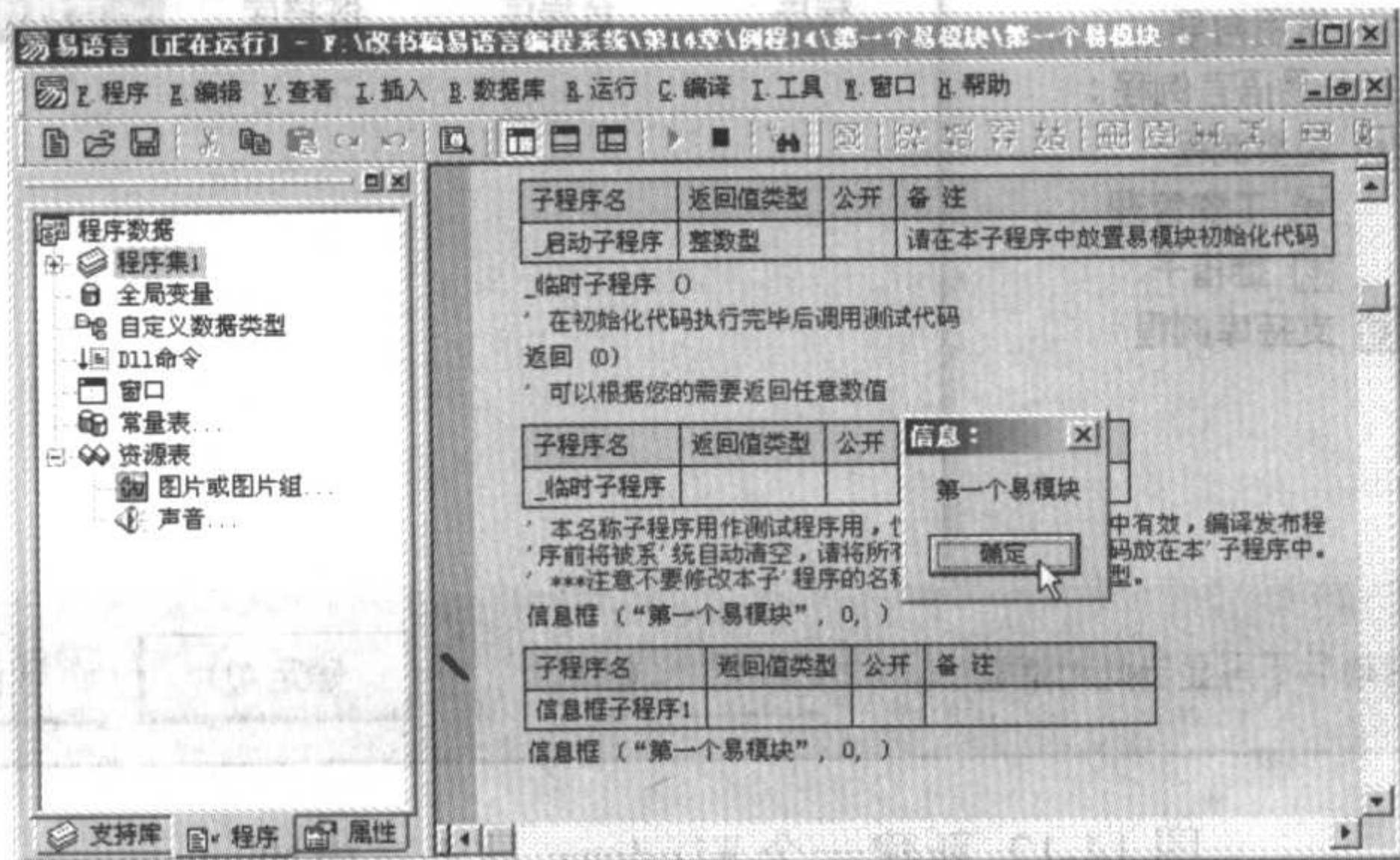


图 14-14 调试运行的结果



4. 如果运行后调试的结果正确, 便可继续编写其他模块子程序, 作为模块接口, 供其他程序调用的子程序, 只需在子程序的“公开”属性打上勾即可。

所有代码被封装在模块中, 模块的使用者只能看到子程序的结构, 无法看到完整的程序代码。

这样, 一个简单的“易模块”就创建完成了。

### 14.3.2 易模块的编译

在易语言中, 易模块创建完成后, 要通过编译才可供其他易程序调用。下面是一个完整的易模块的编译过程。

1. 在“工具”菜单中选择“程序配置”选项。如图 14-15 所示。

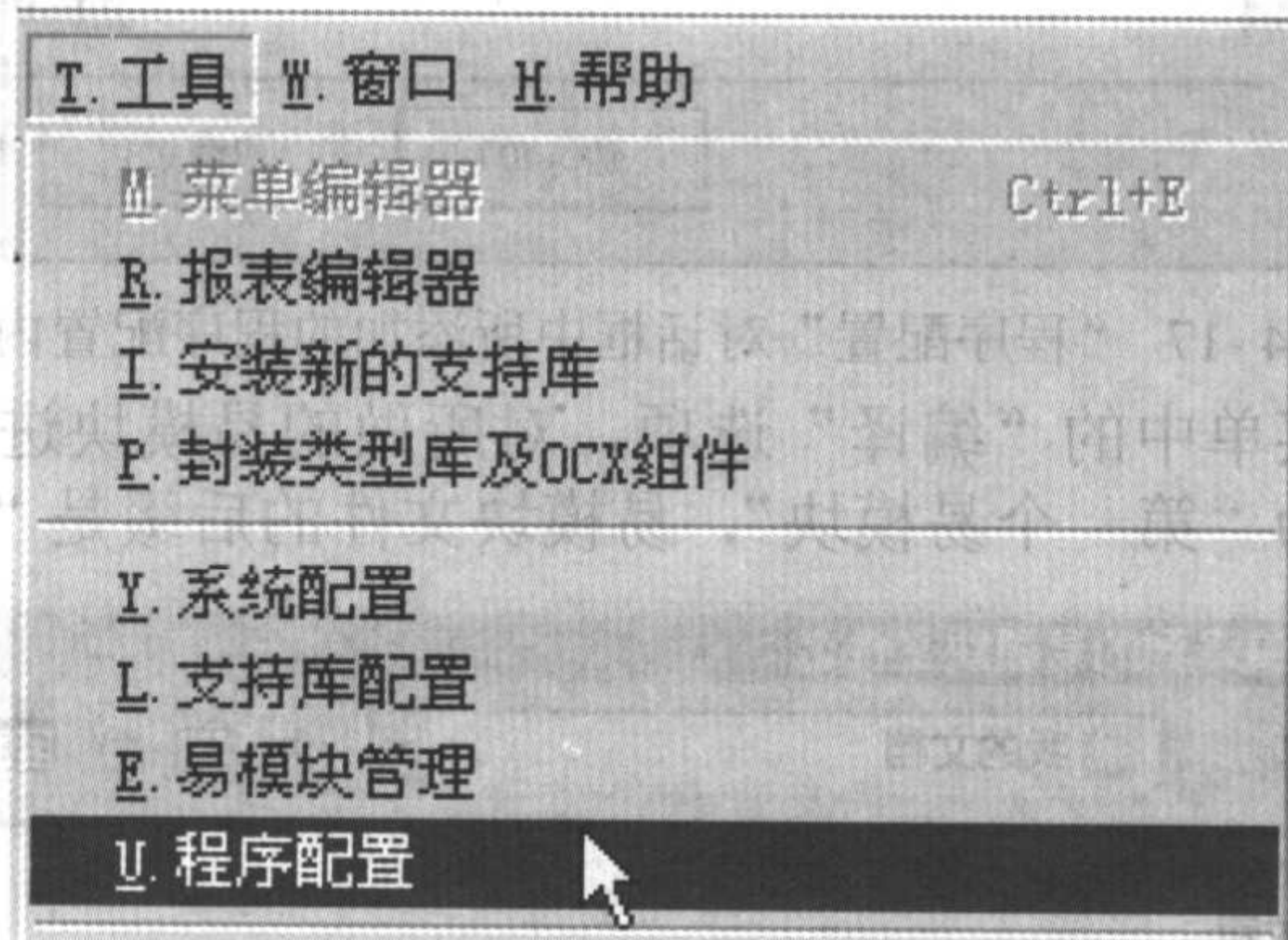


图 14-15 选择“工具”菜单下的“程序配置”

2. 在弹出的“程序配置”对话框中添写相应程序配置的内容。如图 14-16, 图 14-17 所示。

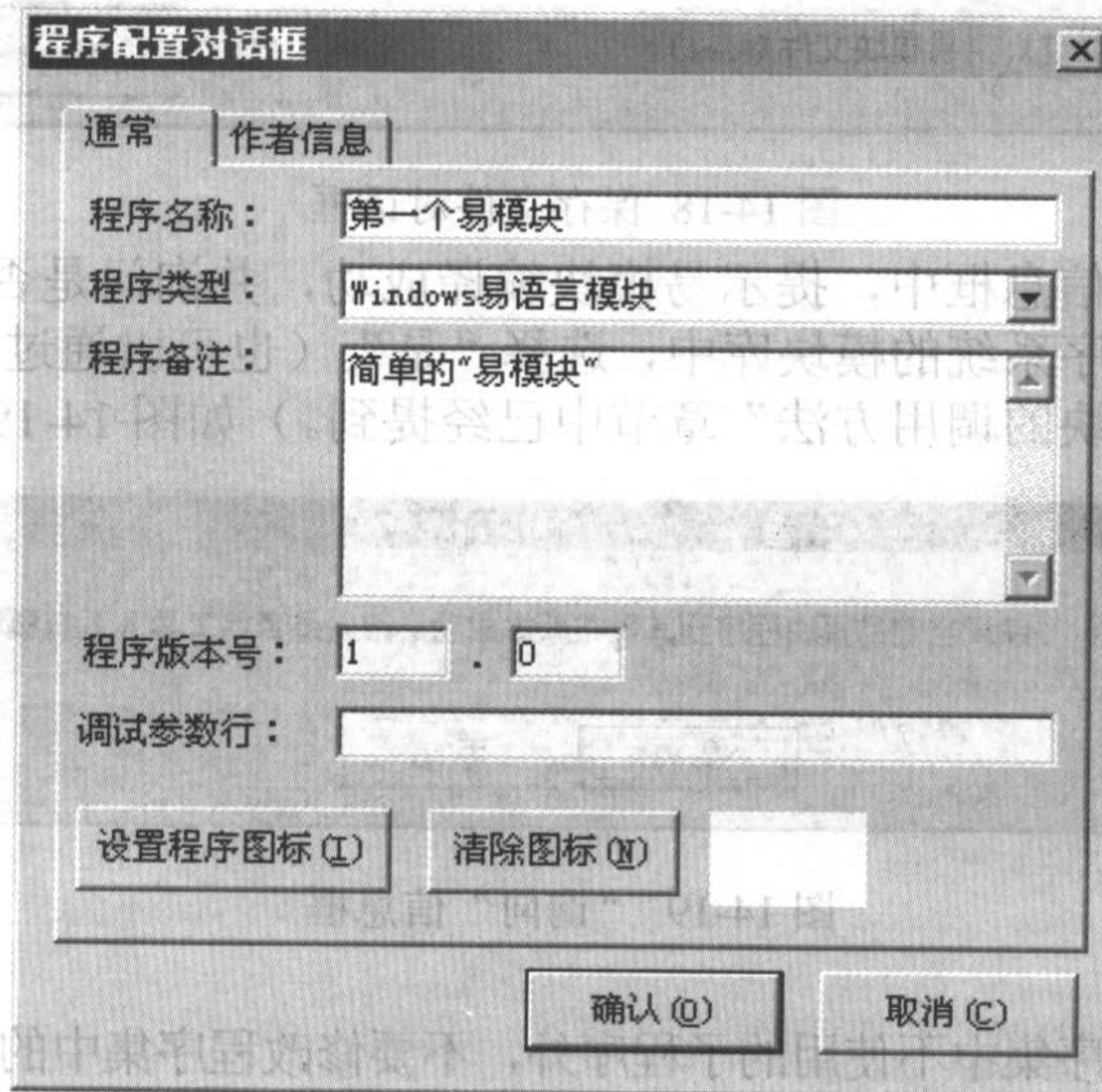


图 14-16 “程序配置”对话框中所添加的程序配置内容



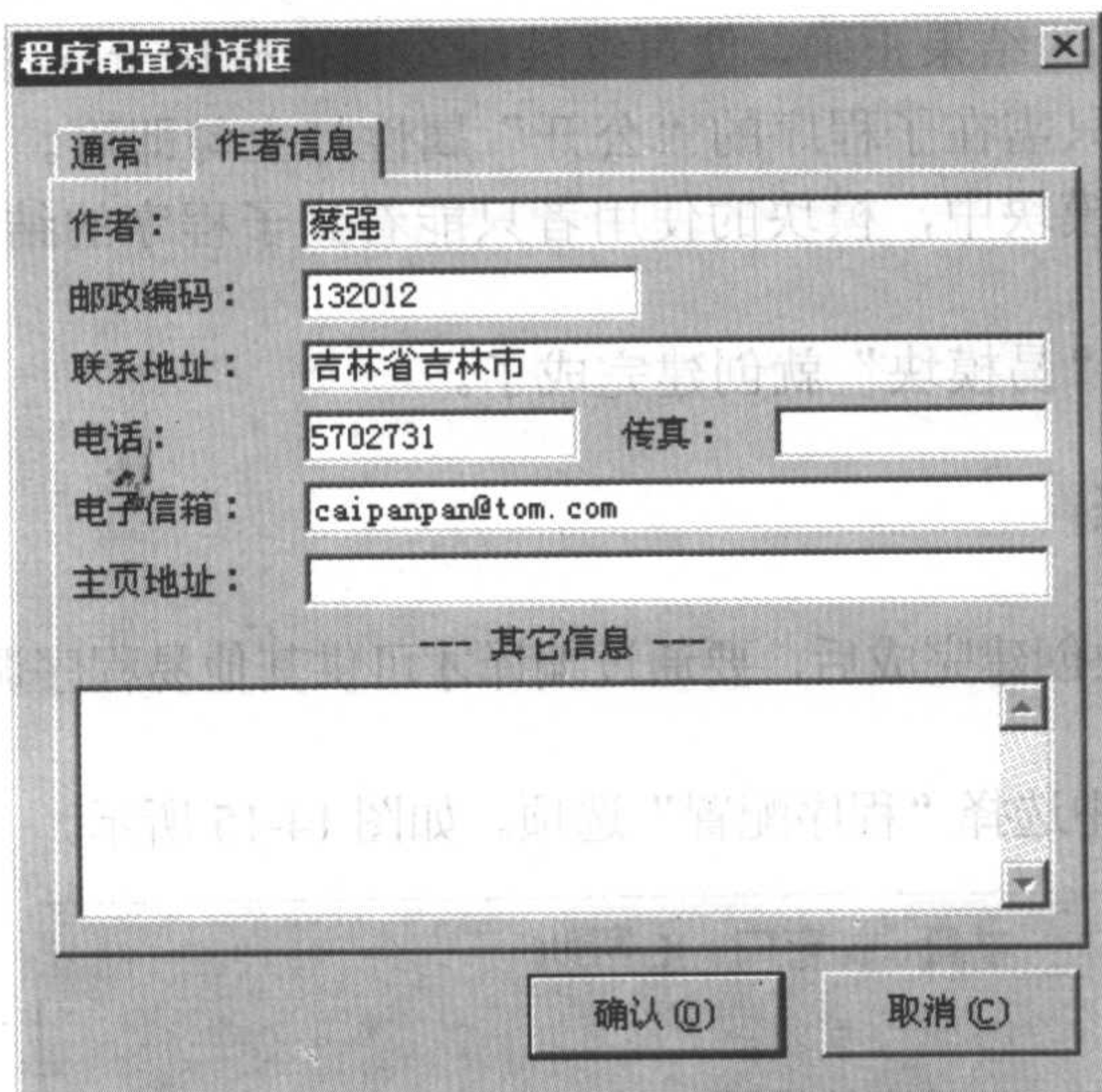


图 14-17 “程序配置”对话框中所添加的程序配置内容

3. 选择“程序”菜单中的“编译”选项，对所做的易模块进行编译。在弹出的保存对话框中，保存文件名为“第一个易模块”，易模块文件的后缀是“.ec”。如图 14-18 所示。

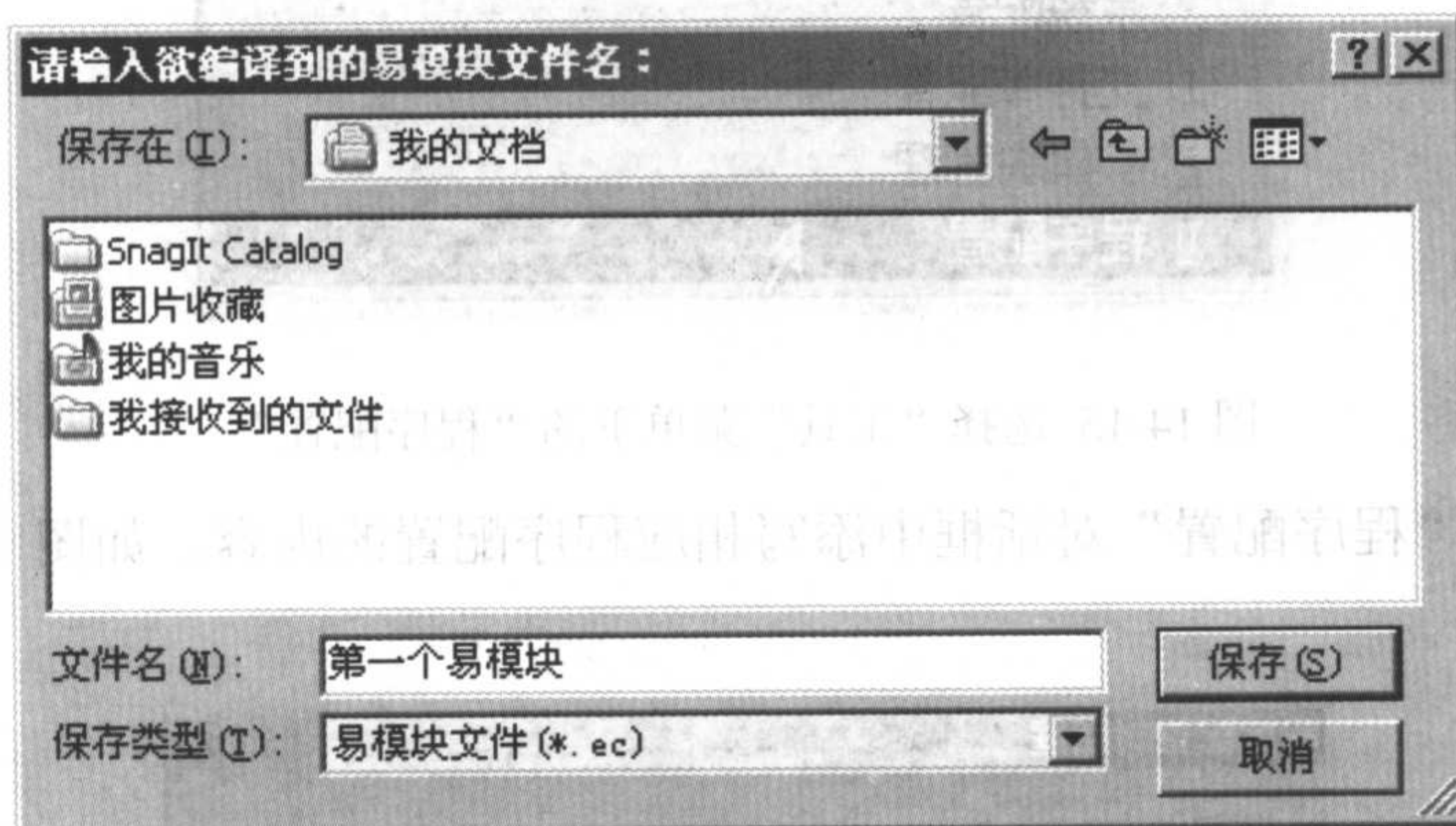


图 14-18 保存文件对话框

在弹出的“询问”信息框中，提示易模块编译成功，并询问是否将当前编译好的易模块程序导入到易语言程序系统的模块库中，选择“是”。（也可以通过“易模块管理”对话框进行导入。在“易模块的调用方法”章节中已经提到。）如图 14-19 所示。

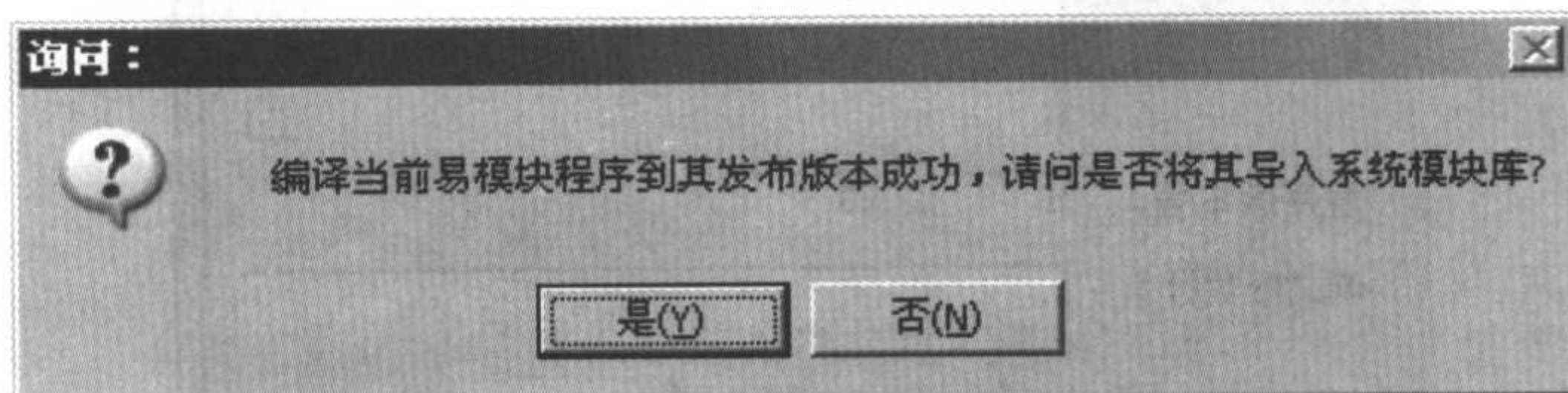


图 14-19 “询问”信息框

### 注意:

1. 除了删除模块程序集中不使用的子程序外，不要修改程序集中的任何地方，否则有可能会编译不通过。



2. 如果想删除对某模块的使用，直接删除为该模块所建立的模块程序集即可。

调用上述模块的过程是：新建一个易程序，导入上述模块，并创建模块程序接口。在“\_启动窗口”中放置一个按钮组件，双击按钮组件，进入按钮被单击事件子程序中，并输入程序代码：“信息框子程序 ()”，试运行，大家可以看到，在点击启动窗口中的按钮后，会弹出显示上述制作好的信息框。这是由于按钮调用了易模块中的子程序完成一个显示信息框的操作。

### 14.3.3 易模块的改写实例

上面是通过一个简单的例子告诉大家易模块的编写方法，下面介绍将易程序中的若干子程序改写成为一个易模块，以便今后在其他程序中使用。

原例程是一个用于处理文字折行的工具，经过改写，使其成为一个易模块，把里面主要的子程序进行封装。然后新建一个“Windows 窗口程序”，导入该易模块，对里面的接口程序进行调用，使其和原例程的运行效果一样。

1. 打开随书光盘所提供的易语言程序“文本折行例程.e”，下面是该例程的运行效果图，如图 14-20 所示。

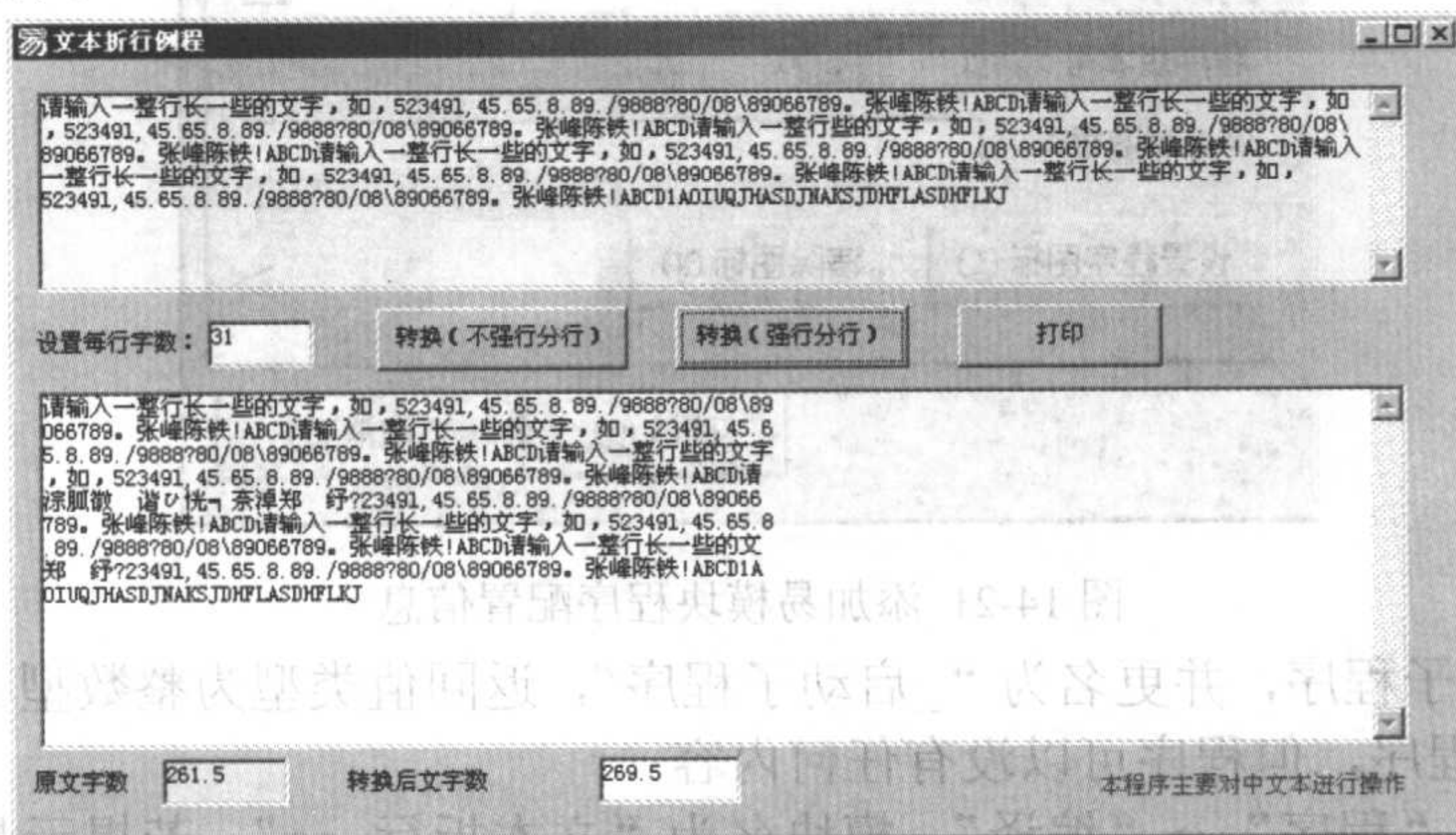


图 14-20 “文本折行例程”的运行效果

“文本折行例程”中关键是使用了一个“文本分行”的自定义子程序，部分代码如下所示：

子程序名	返回值类型	公开	备注		
文本折行	文本型		把一段长文本按你指定的字数分行		
参数名	类型	参考	可空	数组	备注
要处理的文本	文本型				
每行字数	整数型		✓		半角字数，如果为空，将不进行分行操作，直接返回文本内容。
是否强行分行	逻辑型		✓		为真则强行按设定长度分行。为假则保留连续半角数字或字母为一行。默认为假。
是否全角	逻辑型		✓		是否将半角转换为全角，如果为空即不转换
首空字节数	整数型		✓		在每行前面加入空格，如果为空，为0，即不加空格

变量名	类型	静态	数组	备注
操作文本	文本型			
开始位置	整数型			
读取长度	整数型			
开始计算半角	逻辑型			
半角位置	整数型			
结果文本	文本型			





下面大家就跟着本书将这个程序改为模块，将“文本分行”子程序改为模块接口子程序。

2. 打开这个程序后，大家可以另存一个文件名，以保留原来的程序。

3. 将这个子程序的“公开”属性打上对勾。表示作为模块的公开接口子程序，供其他程序调用。

4. 然后通过菜单打开“程序配置对话框”，并添入此模块的相关信息。在“程序类型”下拉列表中选择“Windows 易语言模块”，然后单击“确定”。如图 14-21 所示。

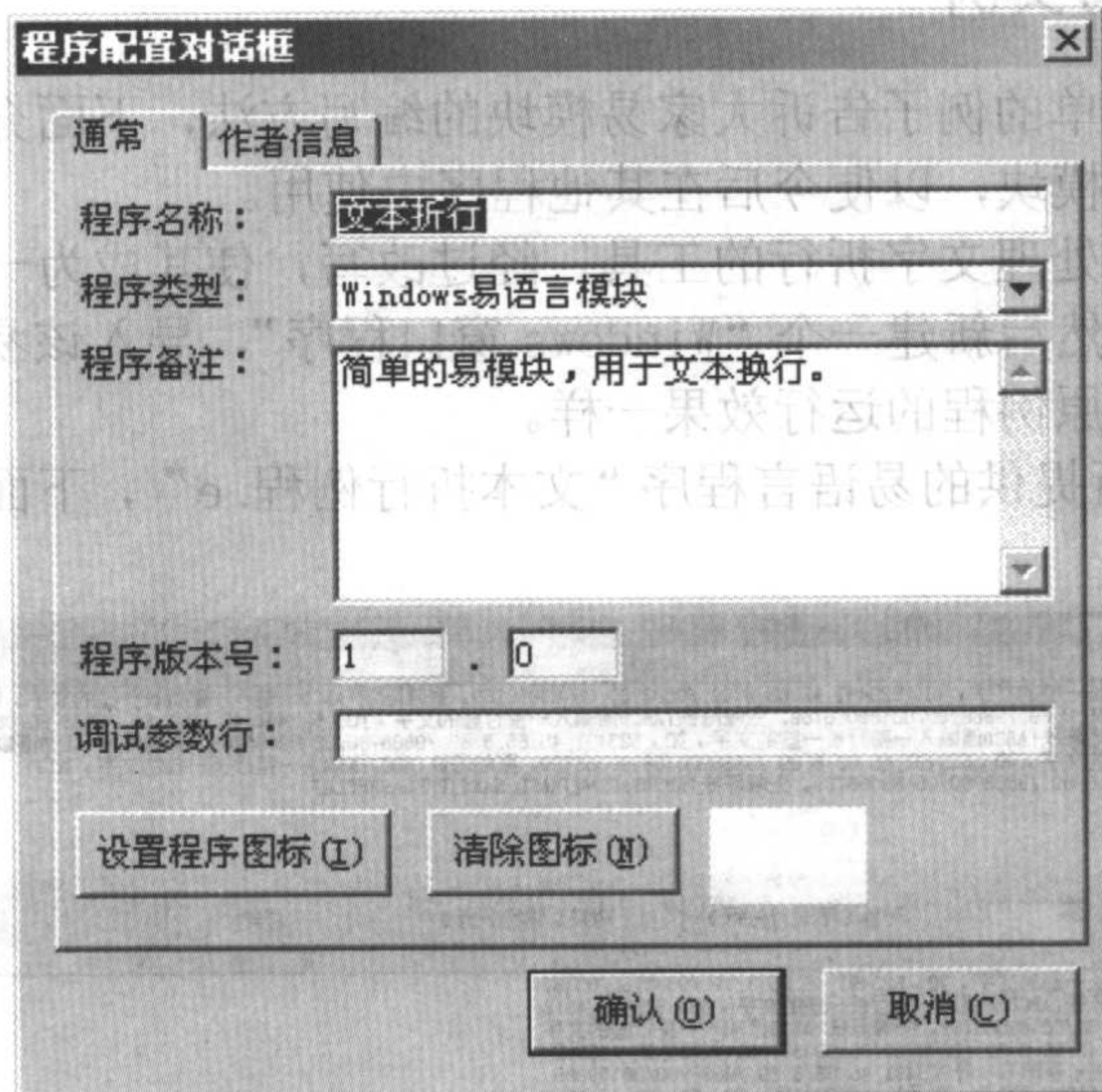


图 14-21 添加易模块程序配置信息

5. 新建一个子程序，并更名为“\_启动子程序”，返回值类型为整数型。这个子程序是易模块必备的子程序，但程序可以没有任何内容。

6. 选择菜单“程序”→“编译”，模块名为“文本折行.ec”。若提示是否导入系统模块库，可以确定导入。这个模块就已建立了。

7. 下面就来测试这个模块，新建“Windows 窗口”程序，导入该易模块。在“\_启动窗口”中，为了与原例程中组件的属性保持一致，可把原例程“\_启动窗口”中的组件拷贝过来。如图 14-22 所示。

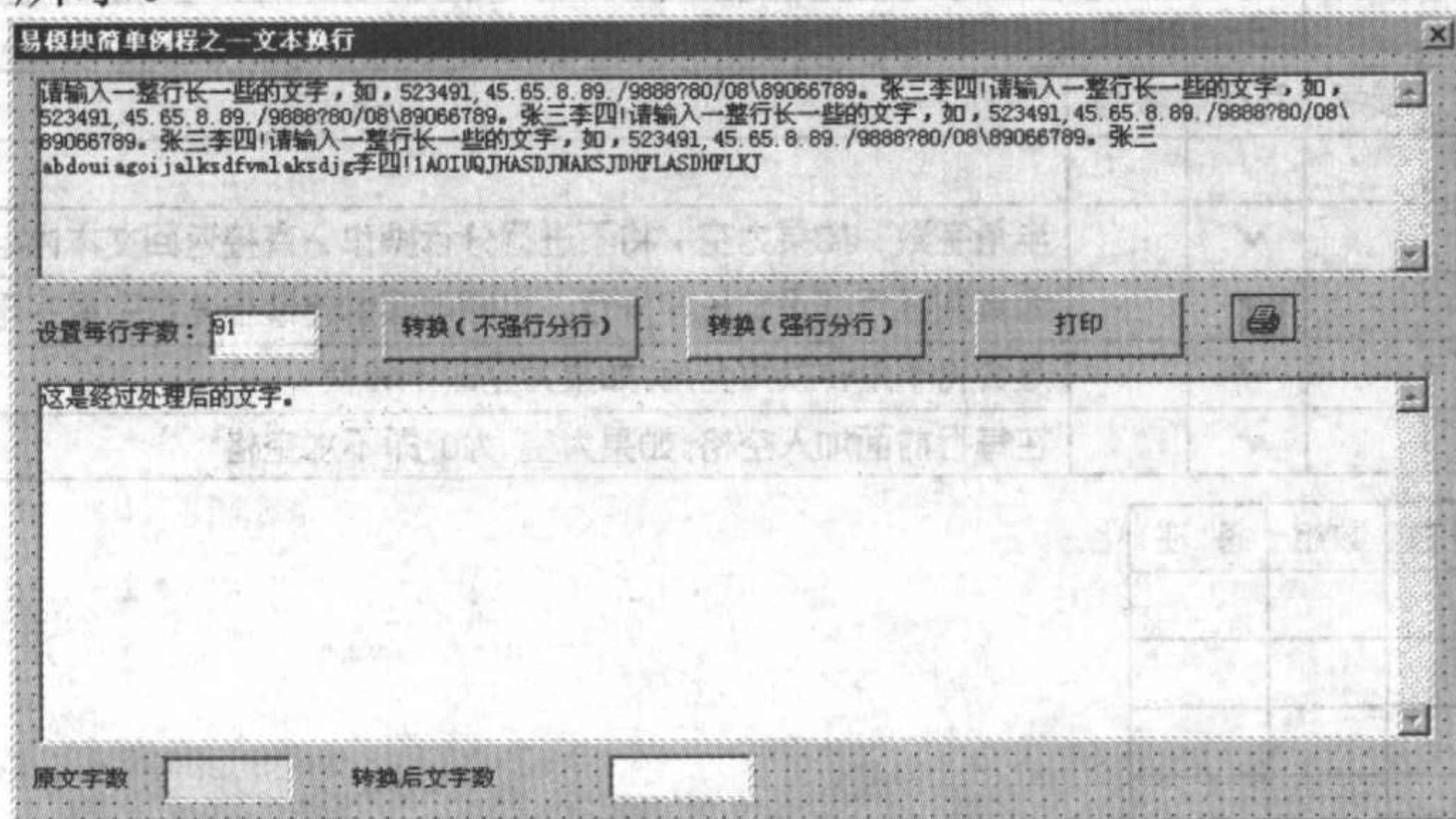


图 14-22 在“\_启动窗口”添加组件



8. 双击“转换按钮”，在“\_转换按钮\_被单击”子程序中输入以下代码：

子程序名	返回值类型	公开	备注
_转换按钮_被单击			

编辑框2.内容 = 文本折行 (到文本 (编辑框1.内容), 到数值 (编辑框5.内容), , , )

在“\_强制转换按钮\_被单击”子程序中输入以下代码：

子程序名	返回值类型	公开	备注
_强制转换按钮_被单击			

编辑框2.内容 = 文本折行 (到文本 (编辑框1.内容), 到数值 (编辑框5.内容), 真, , )

9. 按下 F5 键运行，试着按下“转换按钮”按钮和“强制转换按钮”按钮，会发现在编辑框 2 中显示已经折行的文字，也可以在“设置每行字数”后面设置每行所要显示的文字字数；单击“打印按钮”可对折行后的文字进行打印。如图 14-23 所示。

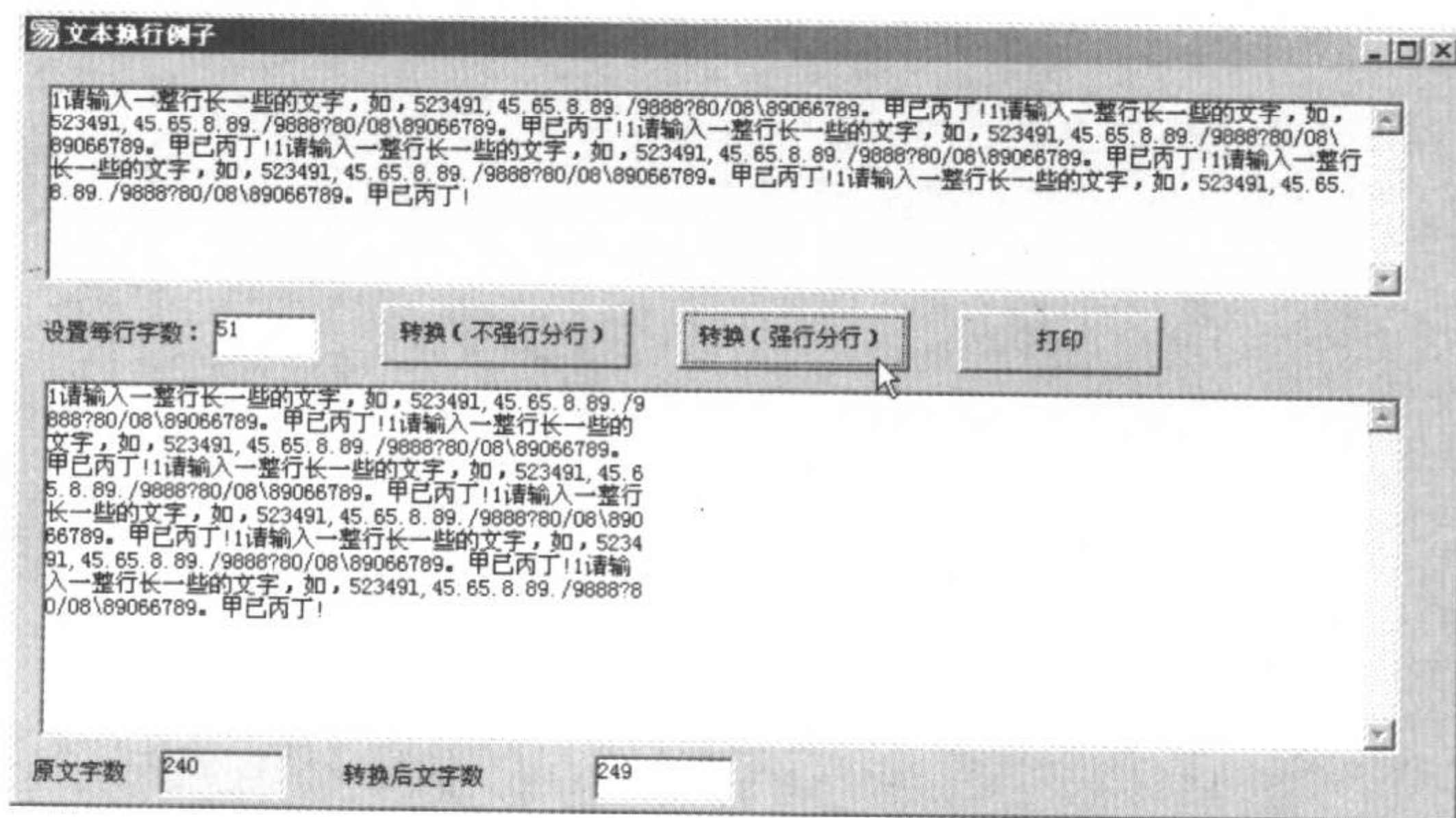


图 14-23 试运行后的效果

至此，整个易模块的改写工作就完成了。以后大家如果在打印时遇到折行的问题，就可以直接使用此模块解决，而用不着重新编写代码了。

改写模块时要注意的就是分清有无使用到其他的程序集变量和全局变量，若有这些类型的变量被使用，改写时就一定要非常注意了。

**注意：**在将普通程序代码改写为易模块时，必须要添加“\_启动子程序”，如果不添加该程序将不能被编译。

## 14.4 本章小结

易模块可为编写程序提供很多方便，如：不用重复写程序代码，别人有的现成模块拿



来即可使用，可以合作开发一个大型的项目，模块源代码不用发布保密性好等优点。

易模块编译后的文件是机器码文件，所以说保密性好。

易模块可以再调用易模块，但嵌套深度建议小于三层。

易语言编译时会将使用到的易模块一同编译到可执行文件中。

大家还可以进行以下练习：

1. 深入理解易模块在程序模块化开发时所起的作用。
2. 学会导入和使用本章所创建的易模块，若其他用户提供的易模块。
3. 试着创建一个求两数和与返回最大数的易模块。
4. 试着将自己将“文本折行例程.e”改写为易模块。
5. 试着将其他程序中的子程序改写为易模块，并编程测试。



## 第十五章 DLL 的编写与调用

### 15.1 DLL 与 API 函数的关系

在第十三章中谈到过 API 函数，大家应该记得每次调用 API 函数前，都必须指明该函数存在于哪个 DLL 文件中。操作系统的标准 DLL 文件中封装了很多底层函数，以供其他编程工具调用。同样，易语言也能用 DLL 文件封装自己编制的函数，供其他编程工具调用，实现了与其他编程工具协作开发一个大型项目的能力。

易模块只适合在易程序中调用，无法供 VB、VC 之类的编程工具调用，而编译成 DLL 文件可以解决此类问题。国外的编程工具对中文的处理实现相对繁琐，使用易语言封装一些中文处理函数，将极大的方便编程人员的工作。

### 15.2 DLL 的开发与编译

从易语言 3.6 版开始，已经能够支持对 DLL 动态链接库的开发，编译出的 DLL 是标准的 WIN32 DLL 文件，可供其他编程语言工具调用，如 VB、VC、Delphi 等。

用易语言编译出的 DLL 文件，需要附带支持库一并提交给用户。

在这里通过一个简单的例程向大家介绍 DLL 的编写方法。此例程通过调用自定义 DLL 文件中的命令来打开一个自定义的信息框。

1. 启动易语言，新建一个“Windows 动态链接库”程序。如图 15-1 所示。

易语言自动创建编写动态链接库的代码设计工作区（程序集）。如图 15-2 所示。

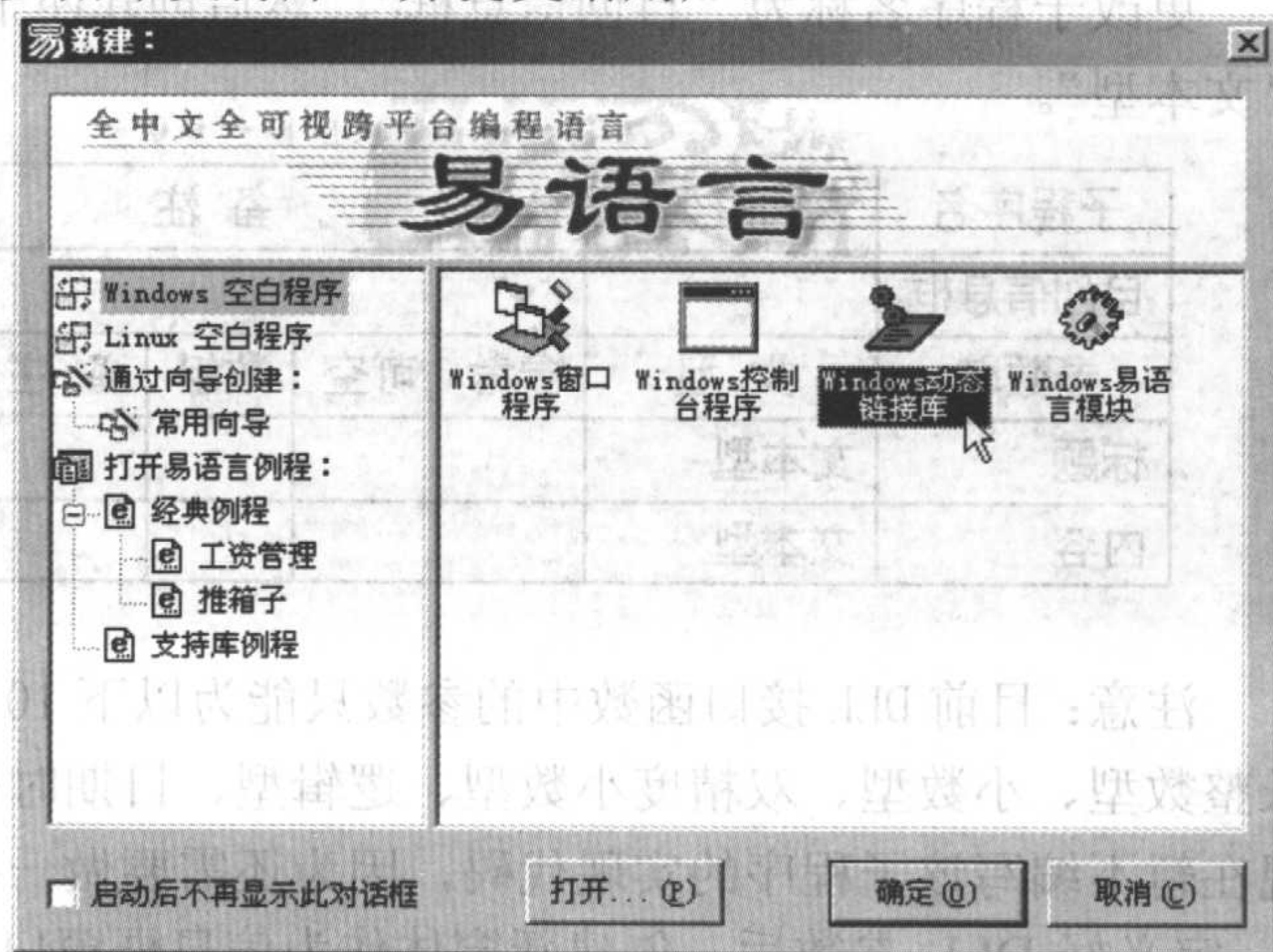


图 15-1 “新建一个 Windows 动态链接库”



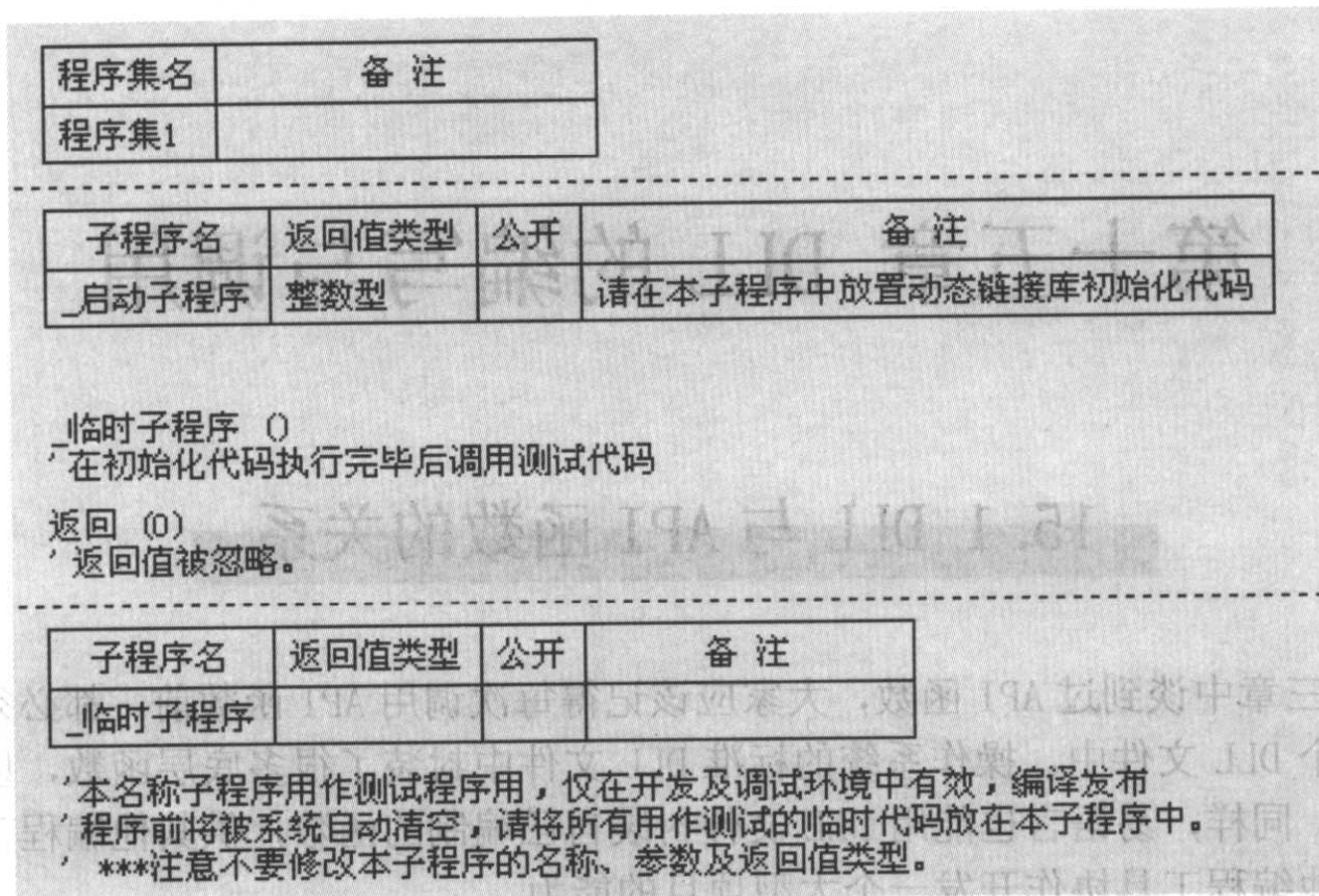


图 15-2 系统自动创建的代码设计工作区

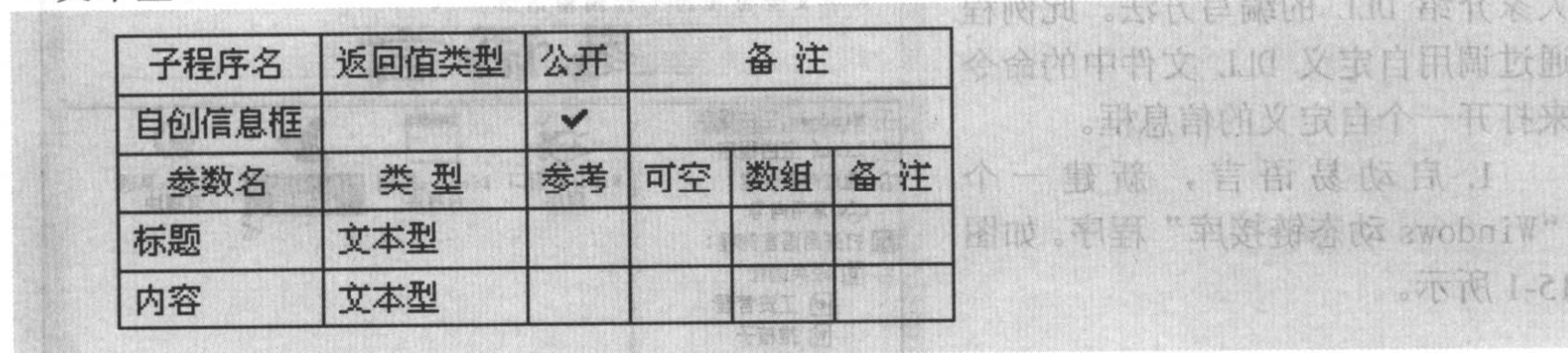
易语言中 DLL 编写方法和易模块的编写方法类似，都需要有对外的公开接口。下面将通过编写一个“自创信息框”来看一下用易语言如何编写和调用 DLL。

编写 DLL 时的公开接口只能由以下方式生成，新建一个子程序，然后把“公开”选中，在 DLL 中任何程序集中选中“公开”的子程序都可作为对外接口使用，如下图 15-3 所示：

子程序名	返回值类型	公开	备注
子程序1		✓	

图 15-3 选中“公开”

更改子程序名称为“自创信息框”，然后创建两个参数“标题”和“内容”，都定义成“文本型”。



注意：目前 DLL 接口函数中的参数只能为以下 10 种之一：字节型、短整数型、整数型、长整数型、小数型、双精度小数型、逻辑型、日期时间型、子程序指针型、文本型。

现在暂不编写该子程序的实现代码，因为还需要做一些准备工作。

2. 定义好 DLL 参数后，创建新窗体作为信息框窗口，在新窗体上添加一个“标签”组件和一个“按钮”组件，并将按钮的标题属性设为“确定”，如图 15-4 所示：



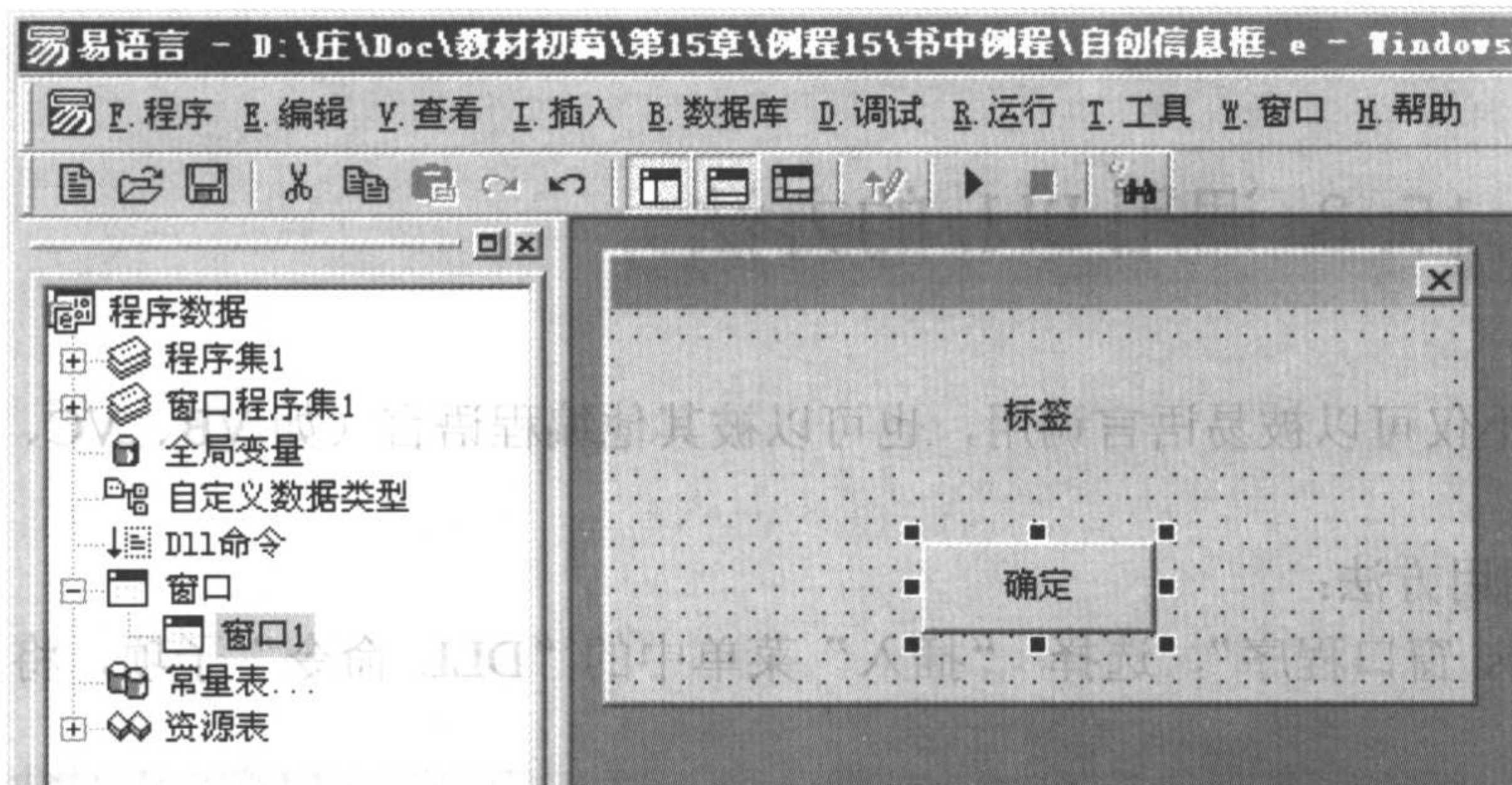


图 15-4 在窗口中添加组件

3. 双击按钮，在“\_按钮1\_被单击”事件子程序中，写入如下代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

窗口1.销毁 ()

4. 编写“自创信息框”子程序的实现代码：

子程序名	返回值类型	公开	备 注		
自创信息框		✓			
参数名	类 型	参考	可空	数组	备 注
标题	文本型				
内容	文本型				

载入 (窗口1, , 假)

窗口1.标题 = 标题

窗口1.标签1.标题 = 内容

5. 到此已经完成“自创信息框.DLL”代码的工作，最后可以选择菜单“编译”→“编译”将其编译为 DLL，文件名为“自创信息框”，编译后的 DLL 文件格式是“.DLL”。如下图 15-5 所示。

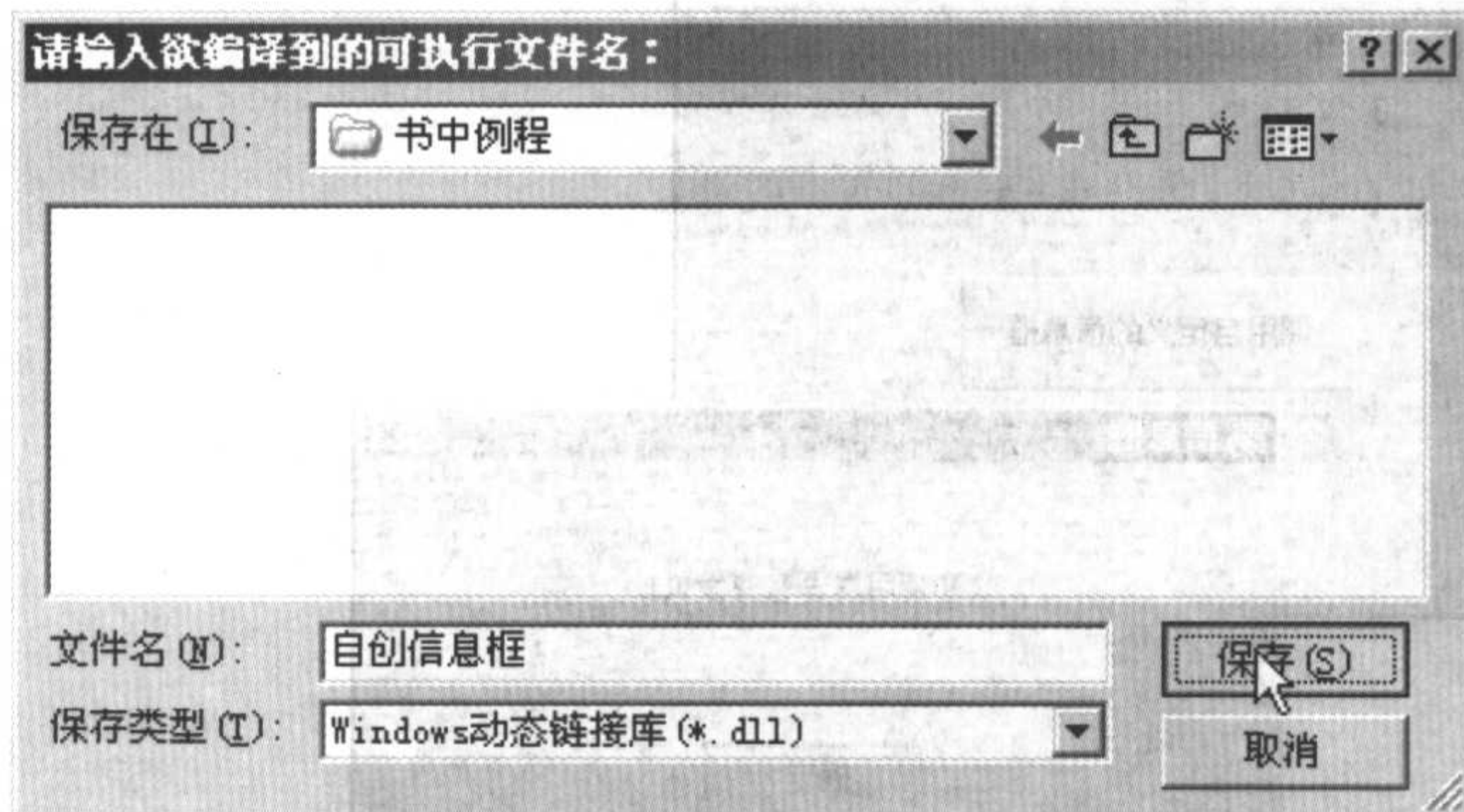


图 15-5 保存编译后的 DLL





## 15.3 调用 DLL 的方法

用易语言编写的 DLL 不仅可以被易语言调用，也可以被其他编程语言（如 VB、VC、Delphi 等）调用。

下面是在易语言中的调用方法：

1. 新建一个“Windows 窗口程序”，选择“插入”菜单中的“DLL 命令”子项，将 DLL 声明加入。

Dll命令名	返回值类型	备 注		
自创信息框				
Dll库文件名				
自创信息框.dll				
在Dll库中对应命令名				
自创信息框				
参数名	类 型	传址	数组	备 注
标题	文本型			
内容	文本型			

注意：“DLL 库文件名”中所指定的 DLL 文件必须位于 EXE 文件所在目录或系统目录，否则运行时将出错。

2. 在窗体上添加一个按钮组件，双击按钮组件，产生“\_按钮 1\_被单击”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

自创信息框（“信息框：”，“欢迎使用易语言开发DLL”）

3. 把此程序保存在与所用到的“自创信息框.dll”文件相同的目录下，运行程序，点击按钮，通过调用 DLL 中的接口函数打开一个自定义的信息框。如图 15-6 所示。

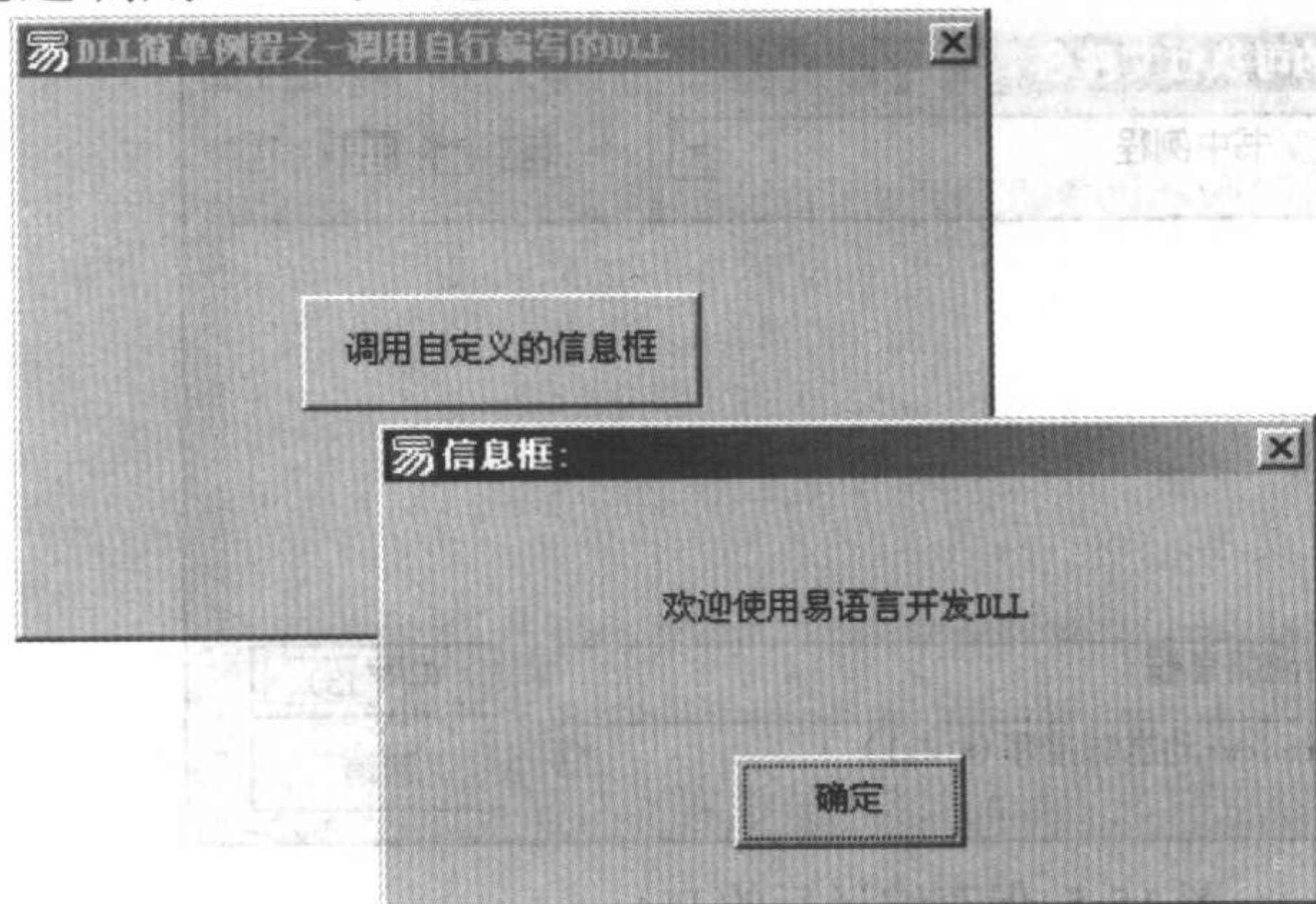


图 15-6 试运行后的结果



注意：

1. 调用 DLL 时的 DLL 对应命令名必须和接口名称完全一样。
2. 接口函数的参数必须是基本数据类型，且不能为字节集型。

上面所编写的 DLL 命令也可以在 VB 中进行调用，新建一个 VB 程序，在窗口添加一个按钮组件，双击按钮组件，添写如下代码（如图 15-7 所示），最后运行即可。

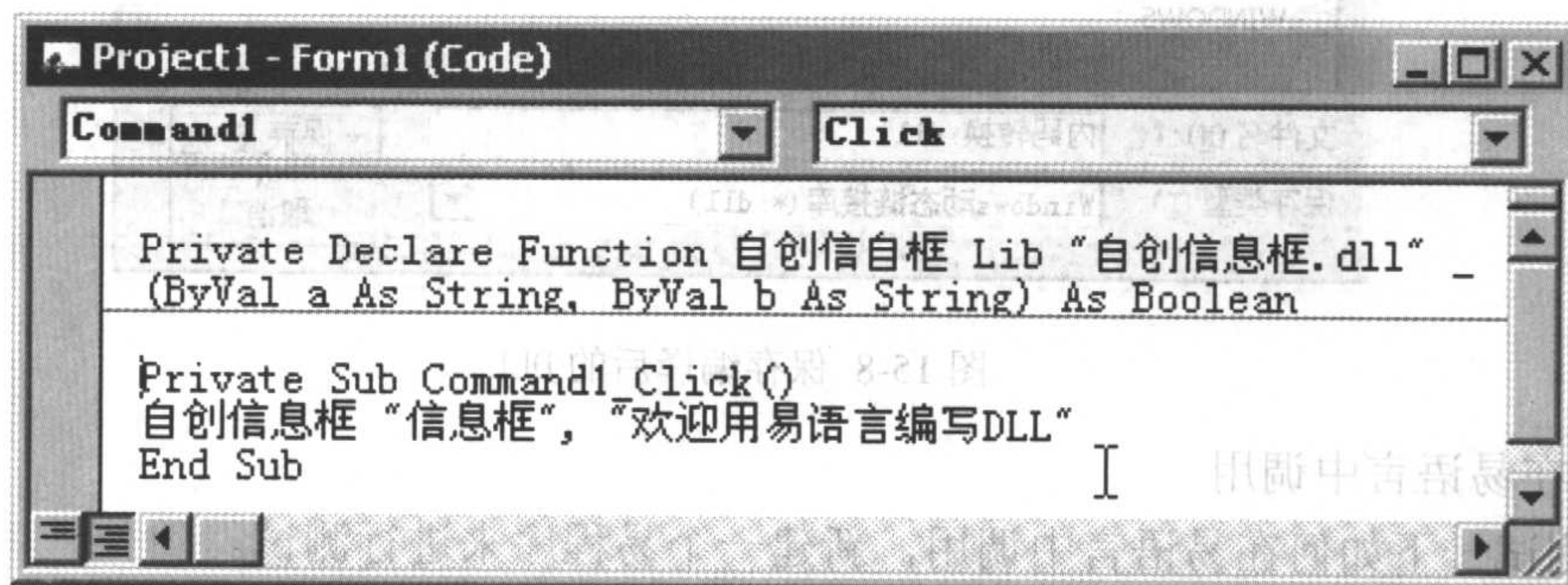


图 15-7 在 VB 中调用 DLL 的代码

## 15.4 DLL 应用实例

由于 VB 不提供简体中文转换为繁体中文的命令，而易语言具有本土化特色的命令，因此可用易语言为 VB 定做一个 DLL，以让 VB 也能得到本土化的功能。

1. 新建一个“Windows 动态链接库”程序，在其中新建一个子程序。将新子程序的名称改为“内码转换接口”，并将子程序的“公开”属性选中。

将光标移到子程序名称上按回车键，可以为子程序增加参数。增加一个名称为“被转换的文本”的文本型参数。

在此子程序中输出一行程序，代码如下：

子程序名	返回值类型	公开	备注		
内码转换接口	文本型	✓			
参数名	类型	参考	可空	数组	备注
被转换的文本	文本型				

返回（内码转换（被转换的文本，#GBK简体到繁体））

2. 选择“编译”菜单中“编译”命令，对此程序进行编译。在这里把它保存在 C 盘根目录下，文件名为“内码转换.DLL”。如图 15-8 所示。



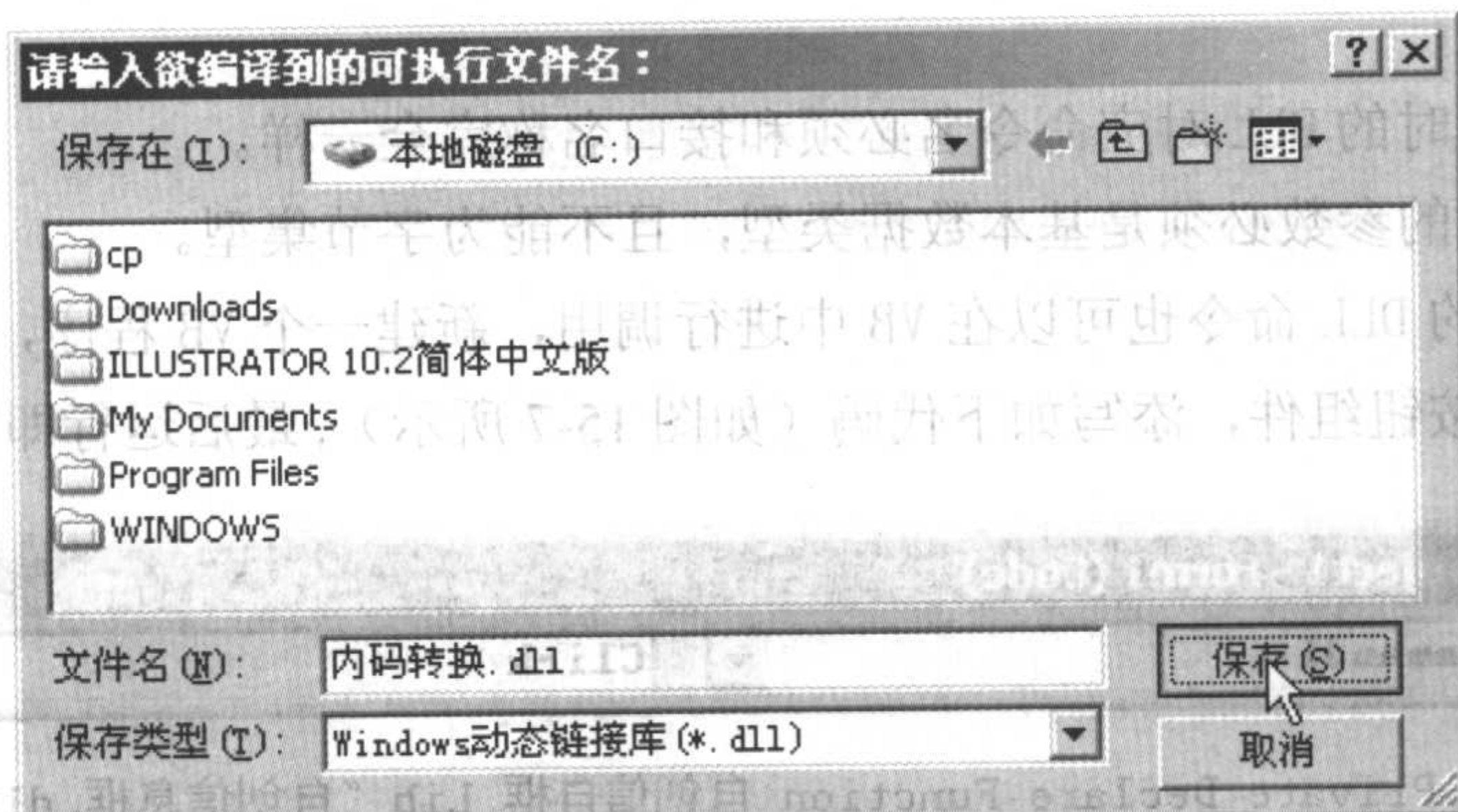


图 15-8 保存编译后的 DLL

### 3. 在易语言中调用

先测试一下如何在易语言中调用，新建一个易程序来进行测试。

在“\_启动窗口”中添加一个标签组件（名称为“标签 1”）、一个编辑框组件（名称为“编辑框 1”）和一个按钮组件（名称为“按钮 1”）。如图 15-9 所示。

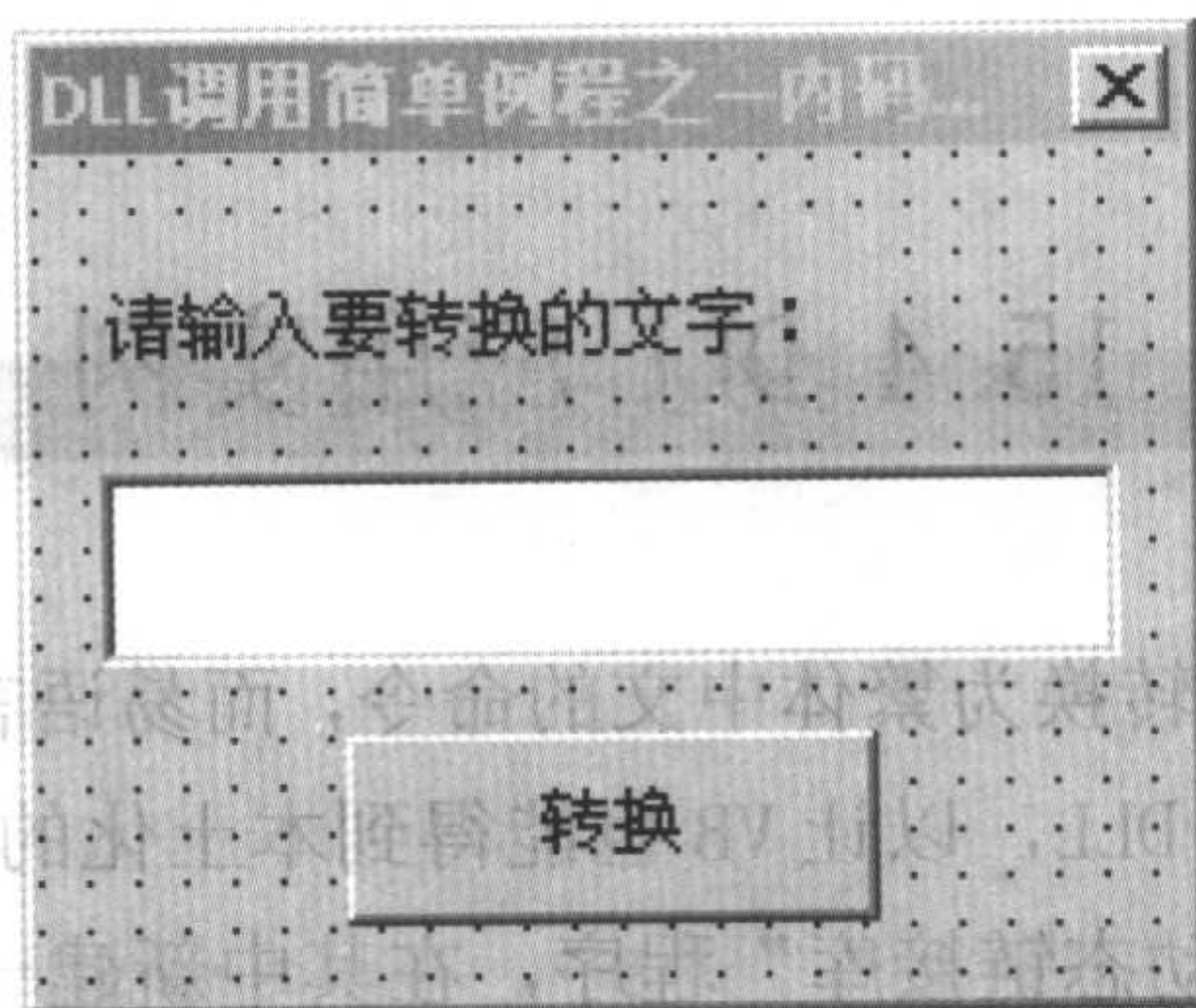
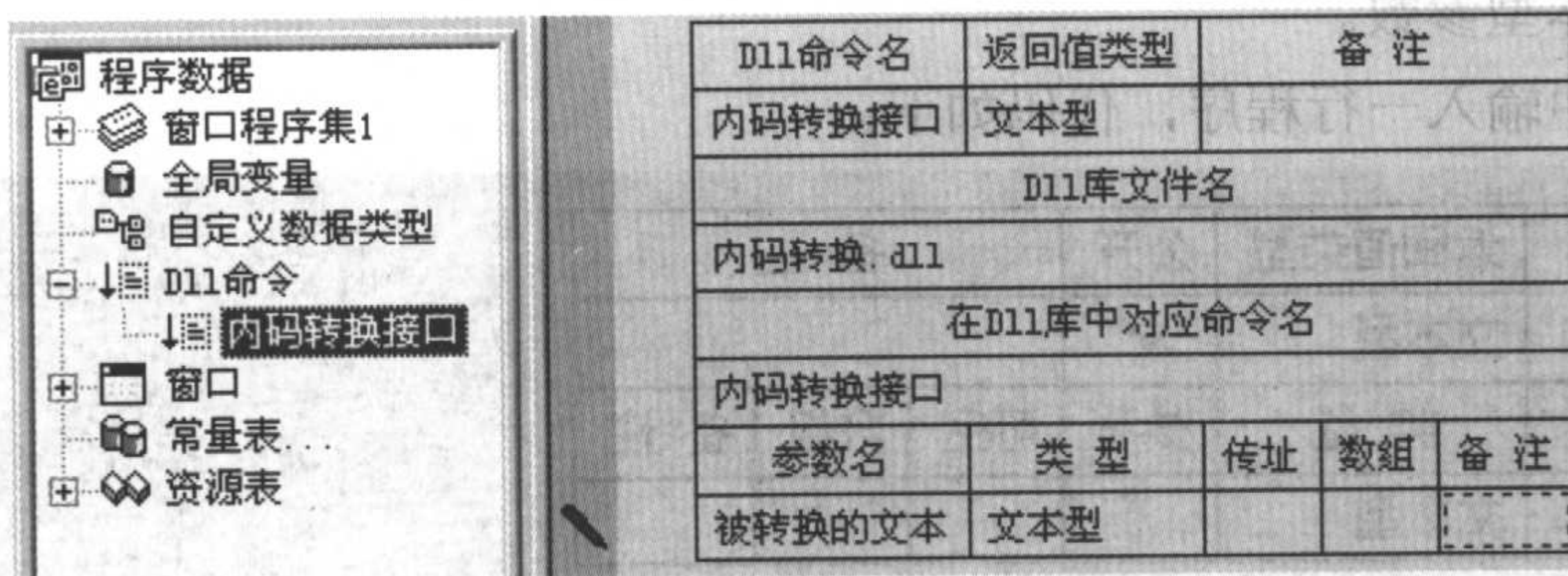


图 15-9 添加了组件的“\_启动窗口”

选择“插入”菜单中的“新 DLL 命令”子项，将 DLL 声明加入。



**注意：**如果所使用的自定义 DLL 文件不在此程序的同一目录，则“DLL 库文件名”中必须填写该 DLL 文件的完整路径。但把路径写入程序中是一个不好的习惯。最好的选择还是把 DLL 文件拷贝到 EXE 所在目录，或系统目录。

在“\_启动窗口”中添加一个按钮组件，双击该按钮组件，产生“\_按钮 1\_被单击”事



件子程序，然后输入以下程序代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

编辑框1.内容 = 内码转换接口 (编辑框1.内容)

4. 试运行一下，看看运行的结果，在编辑框中输入欲转换的中文文本。如图 15-10 所示。

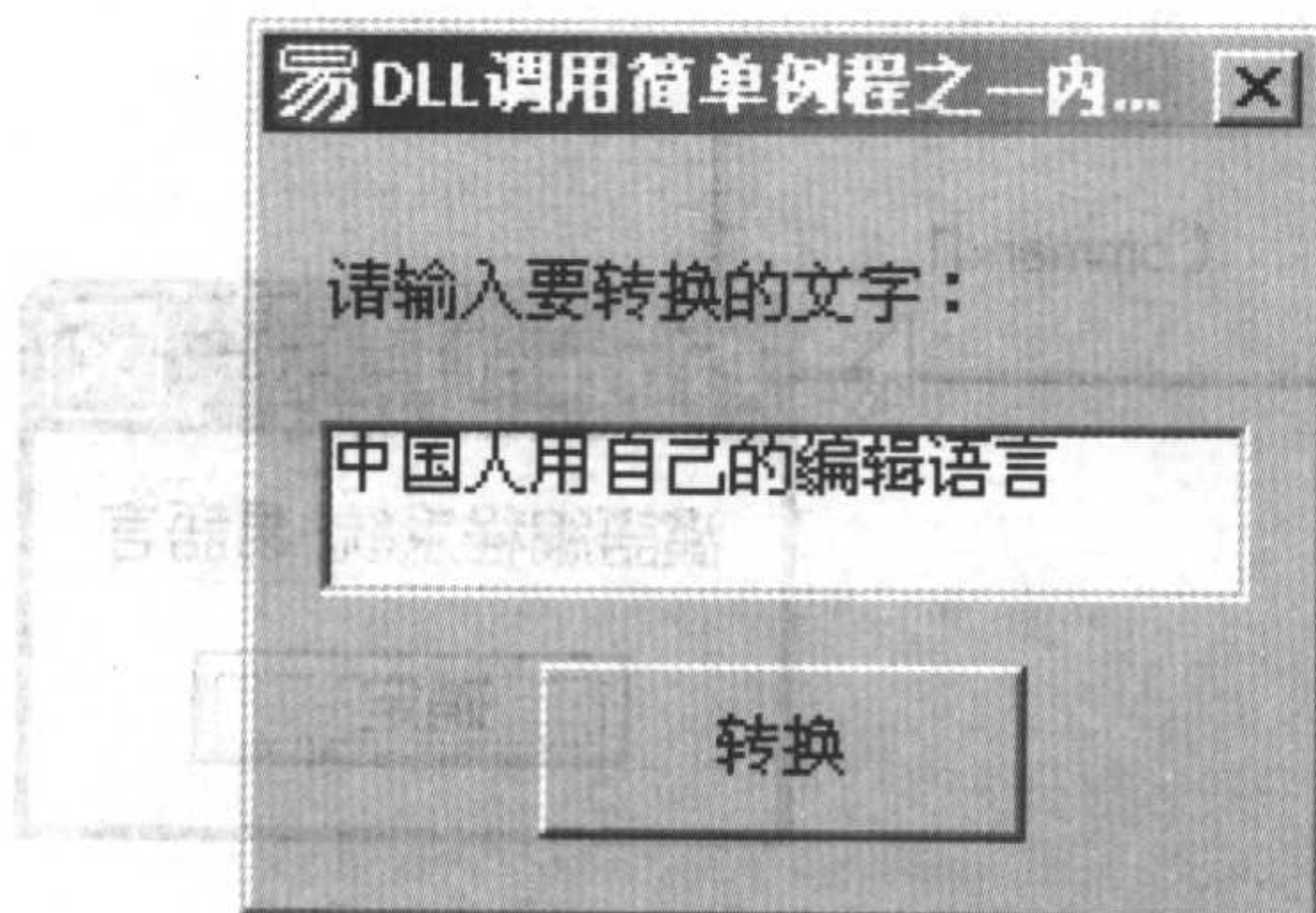


图 15-10 输入文本

点击“转换”按钮后，编辑框中会显示该文字转换后的繁体中文。如图 15-11 所示。

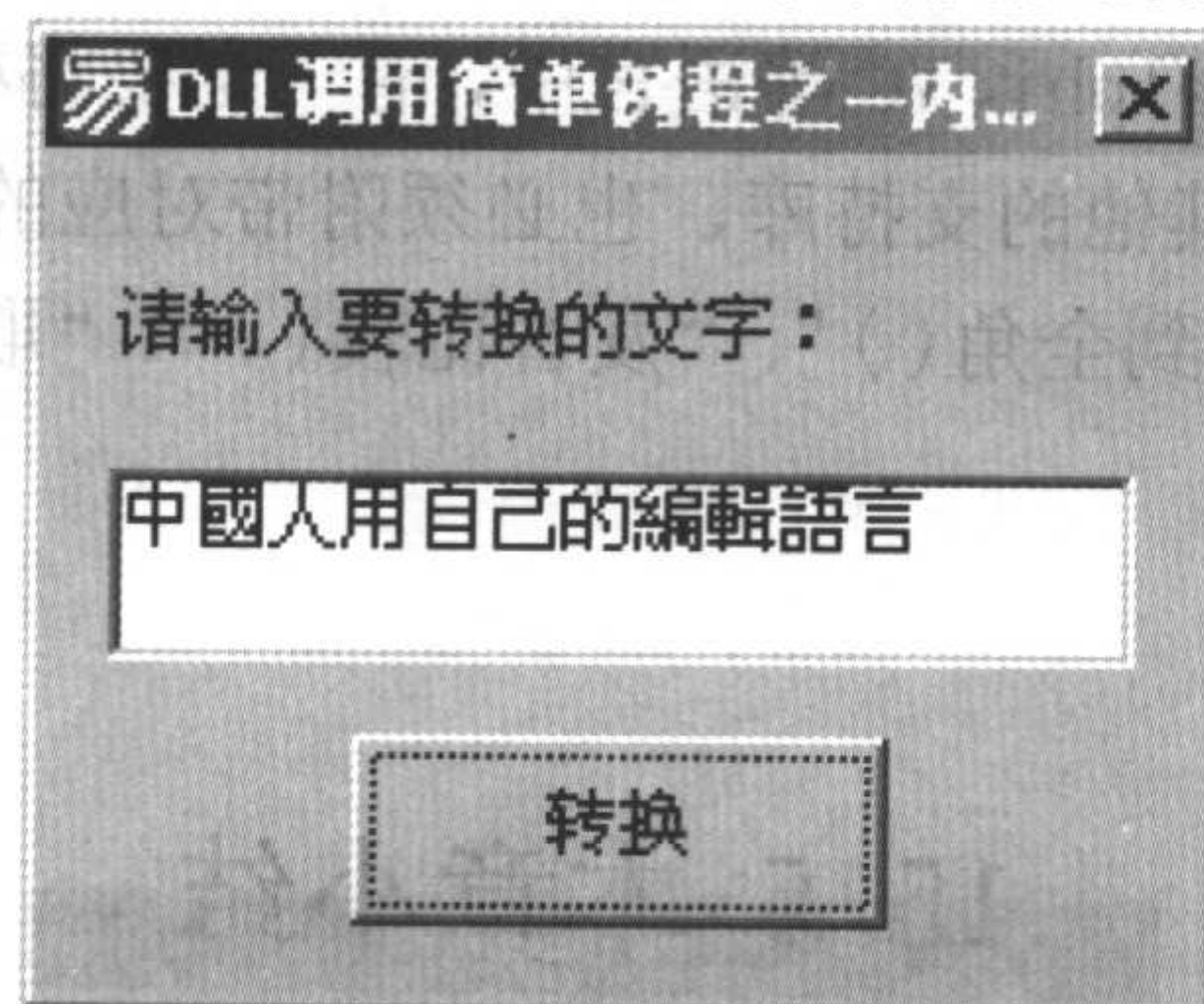


图 15-11 转换后的繁体中文

5. 在 VB 中调用 DLL

新建一个 VB 程序，在窗口添加一个按钮组件，双击按钮组件，填写如下代码。如图 15-12 所示。

```

Command1 Click
Private Declare Function 内码转换接口 Lib "c:\内码转换.dll" _
    (ByVal text1 As String) As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
    (ByVal pDst As Any, ByVal pSrc As Any, ByVal ByteLen As Long)

Private Sub Command1_Click()
    Dim texts, buf As String
    Dim ptr As Long

    texts = "汉语编程系统-易语言"
    ptr = 内码转换接口(texts)

    buf = Space$(Len(texts) * 2)
    CopyMemory buf, ptr, Len(texts) * 2
    MsgBox buf
End Sub
    
```

图 15-12 在 VB 中调用 DLL 的代码





其中，声明了两个 DLL，除易语言提供的 DLL 外，CopyMemory 函数是用于从 DLL 传送过去的内存地址中取得内容。在设置文本长度时，也可以设置大一些。

运行后的结果如图 15-13 所示。

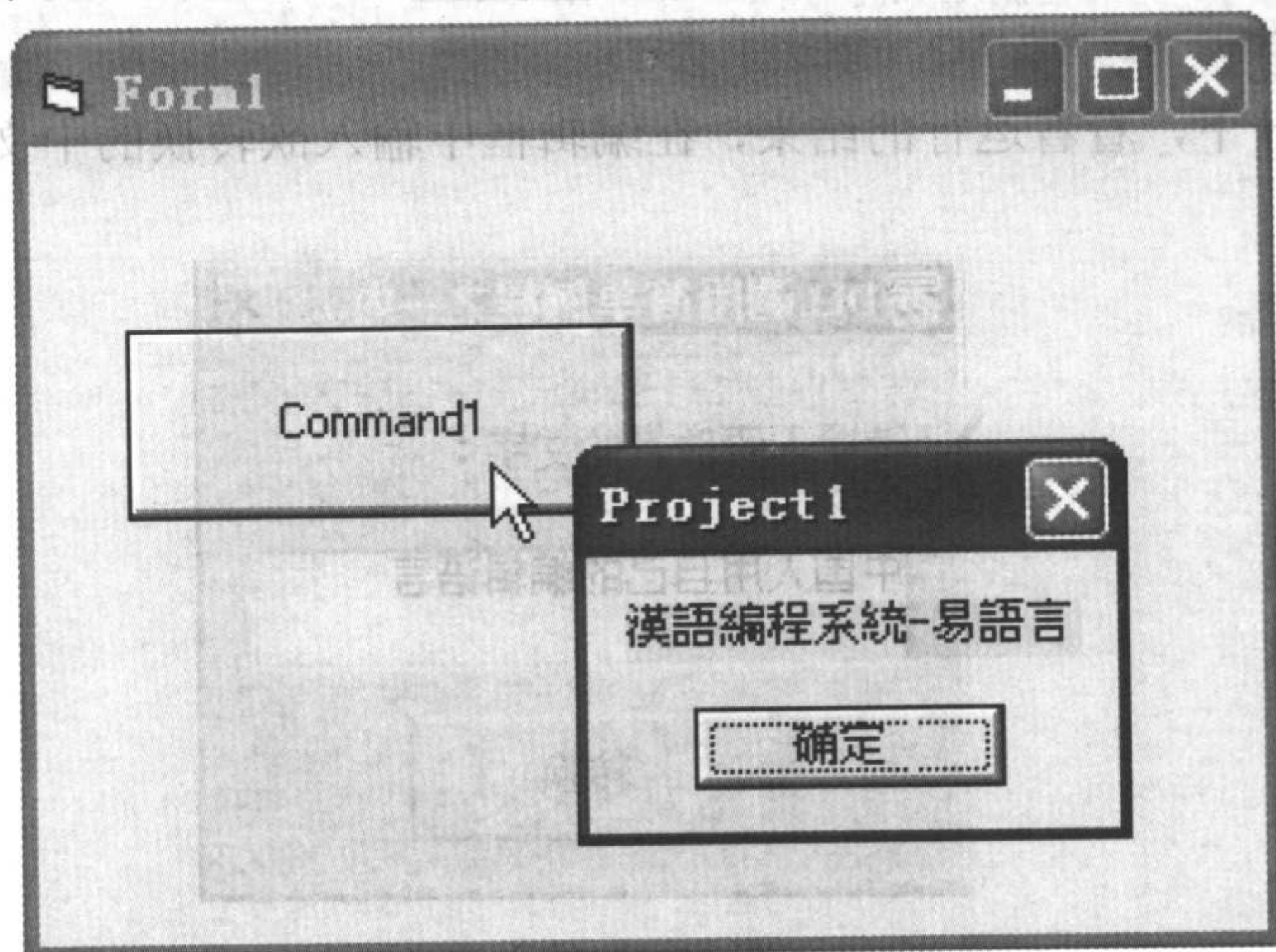


图 15-13 在 VB 中调用 DLL 的运行效果

如果此 DLL 应用到 VB、Delphi、VC 等语言中，请一定要将核心支持库文件“krnl.n.fnr”随程序一同发布。如果用到其他的支持库，也必须附带对应的易语言支持库一同发行。

其他易语言的命令如：“到全角()”、“发音比较()”、“取拼音()”、“数值到大写()”等，均可以这样实现。

## 15.5 本章小结

在易语言中，可以创建标准的动态链接库文件(DLL)供其他编程语言调用。同时，易语言也可以调用由其他编程语言编写的动态链接库，从而达到了编程功能互通的目的。

易语言提供编写 DLL 的功能使得易语言与其他语言共同合作开发一个大型项目成为可能。

大家可以进行以下练习：

1. 了解 DLL 与 API 之间的关系，熟练掌握 DLL 文件的编写与调用。
2. 仔细练习书中所讲解的 DLL 的应用例程，尝试用易语言、VC、Delphi 等编程语言对自定义 DLL 进行调用。
3. 尝试用“到全角()”、“发音比较()”、“取拼音()”、“数值到大写()”等命令编写一个 DLL，并在 VB 中调用。（具体代码可参见本书配套光盘中例程）
4. 将折行打印的主要子程序制作成一个 DLL 使用。



## 第十六章 OCX 组件与类型库

在计算机系统中，已经存在很多现有功能组件和功能接口，譬如微软公司用于扩展其 VB 编程语言功能的 OCX 组件和 TypeLib 类型库、用作充分利用 Windows 操作系统性能的应用程序功能接口（API）、用作与其他外部对象和应用程序交流功能的 COM 协议，等等。易语言支持对所有这些资源的使用。

### 16.1 OCX 组件

OCX 是一种组件，它的名字来源于它的文件扩展名“.ocx”。有大量的计算机软件生产商提供用于共享或商业的 OCX 组件供选择，各种 OCX 组件能完成各种不同的功能。对于使用 OCX 组件只需要将 OCX 组件插入到应用程序中去，并相应的填写 OCX 组件的接口。例如，如果想在 PowerBuilder 应用程序中使用一个 OCX，可以将此 OCX 插入到 PowerBuilder 窗体或数据窗体中的一个 OLE 组件中。

#### 16.1.1 OCX 的安装

在这里以一个“Media Player 播放器控件”为例，来一同看一下易语言中 OCX 控件的安装。

1. 选择易语言菜单“工具”→“类型库或 OCX 组件->支持库”子项，打开安装组件对话框。如图 16-1、图 16-2 所示。

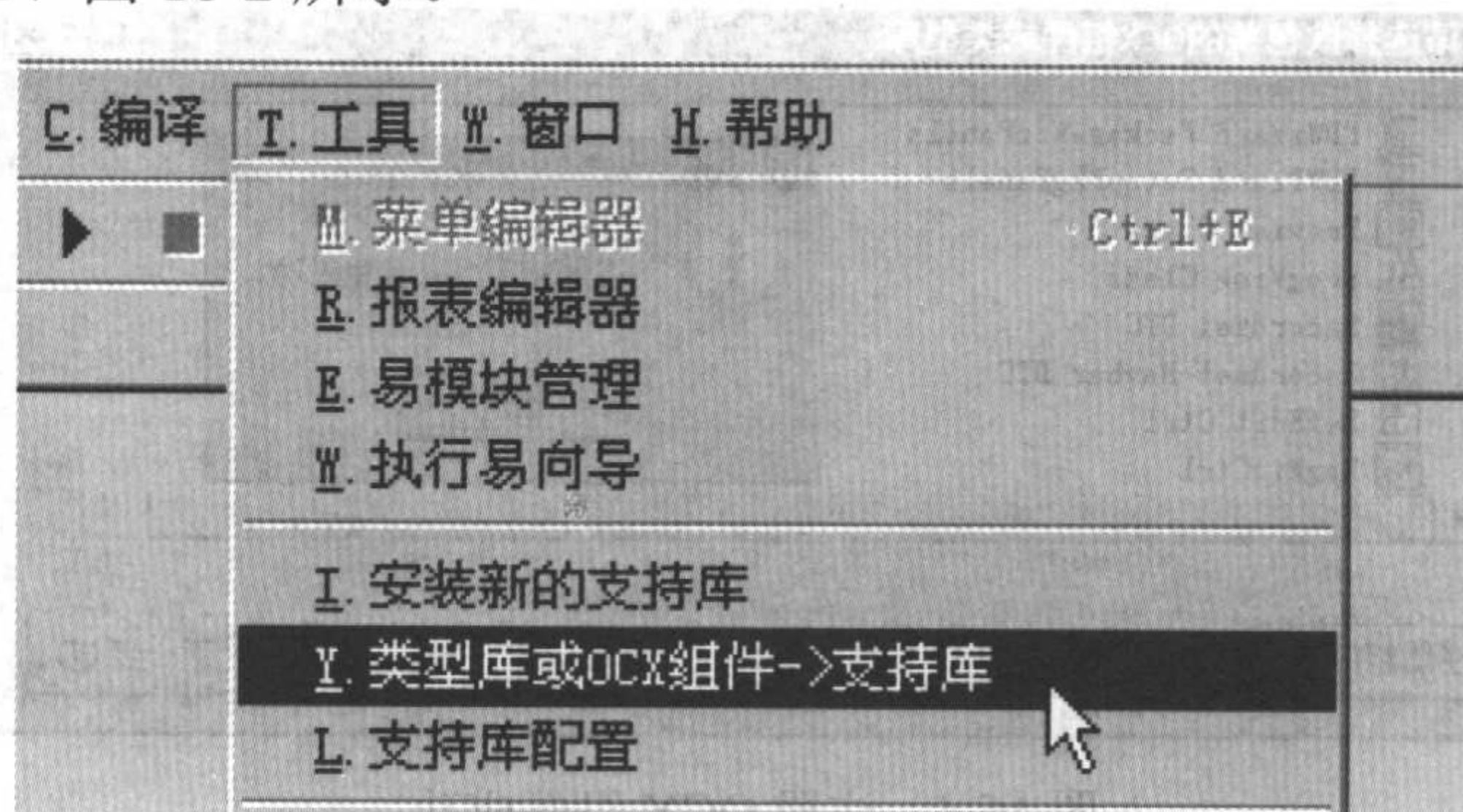


图 16-1 打开“OCX 组件包装器”



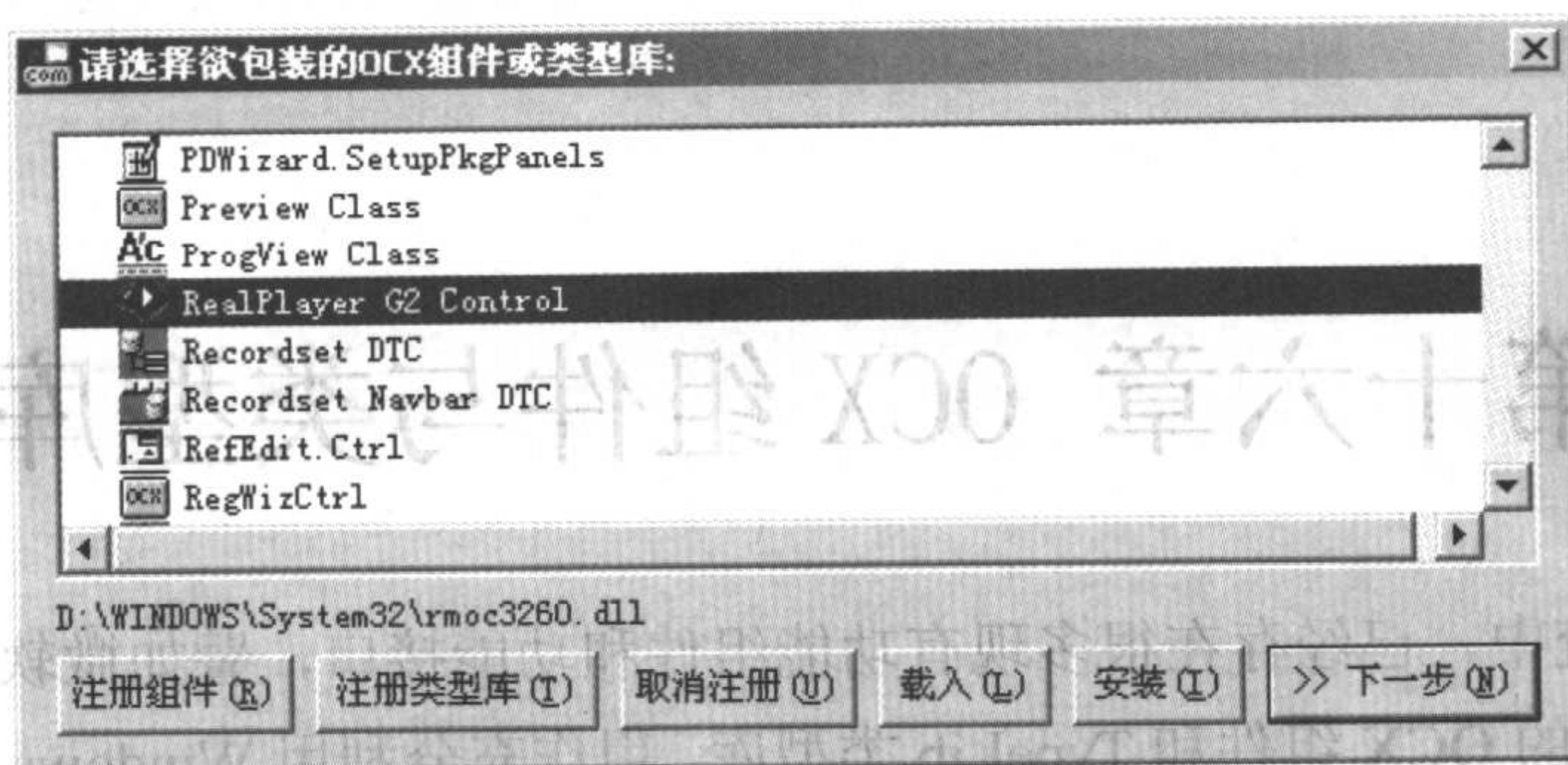


图 16-2 组件安装对话框

图 16-2 中，系统中已安装“RealOne Player”播放器，组件安装工具会将组件添加到组件列表中，省去注册组件的步骤。

2. 如果系统中没有安装“RealOne Player”播放器，首先必须注册光盘中提供的“rmoc3260.dll”文件。将“rmoc3260.dll”文件保存到本地磁盘，点击“注册组件”按钮，选中要注册 OCX 组件的“rmoc3260.dll”文件，点击“打开”按钮，组件被添加到组件列表中并提示。如图 16-3、图 16-4 所示。

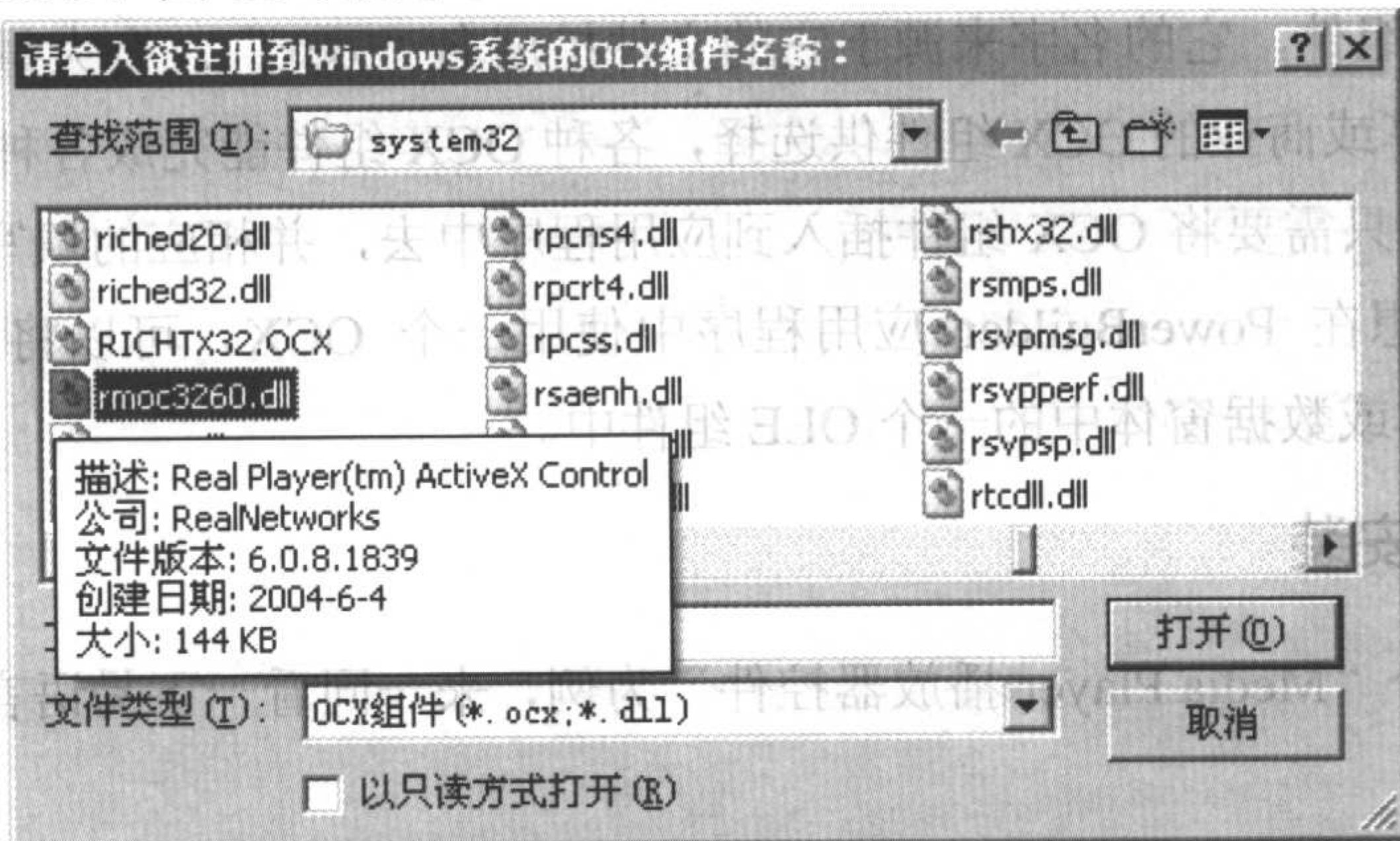


图 16-3 注册 RM 组件

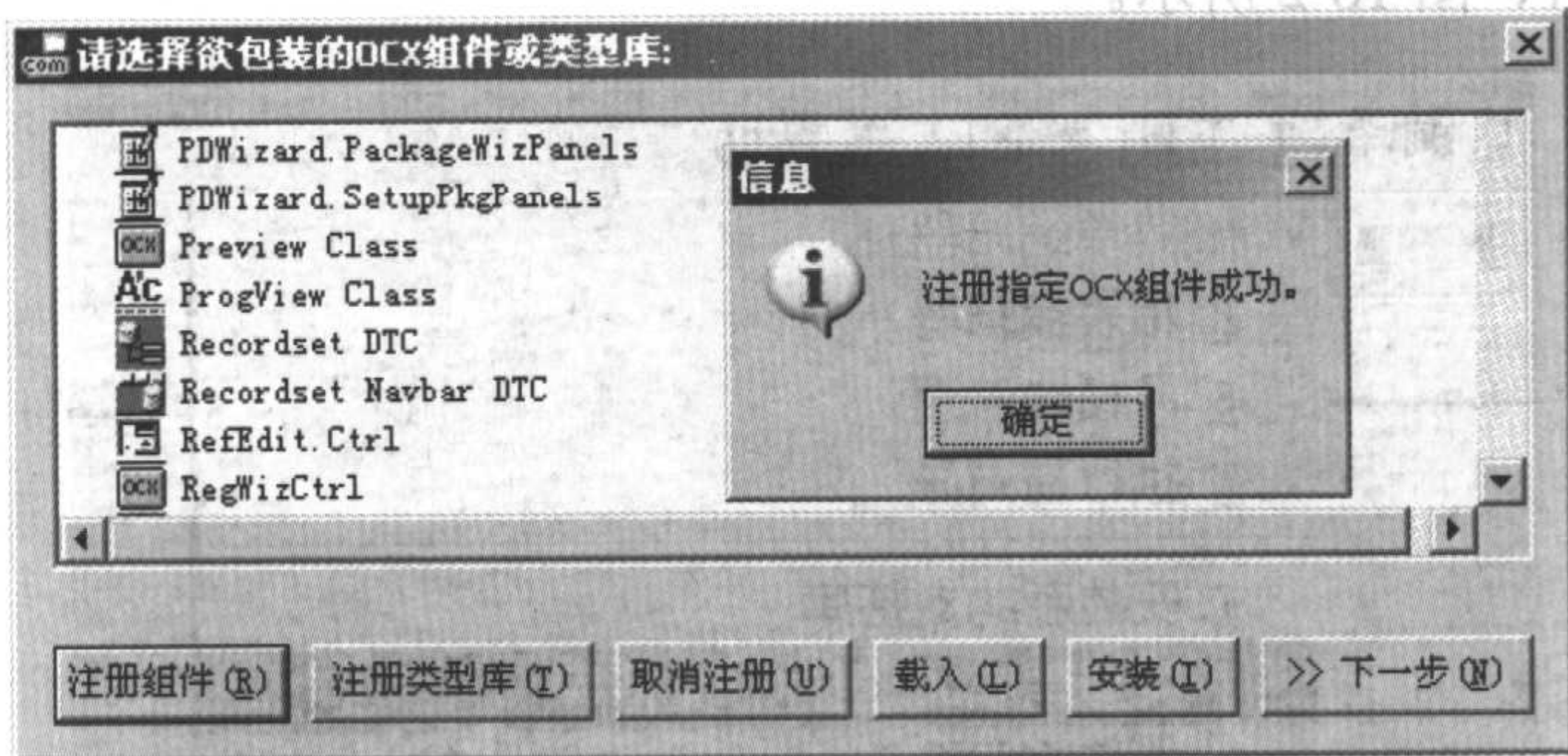


图 16-4 注册 OCX 组件成功

3. 选择新添加的“RealPlayer G2 Control”组件，单击“下一步”按钮，选择提示窗口中的支持库名称后，单击“保存”按钮保存注册信息。如图 16-5 所示。



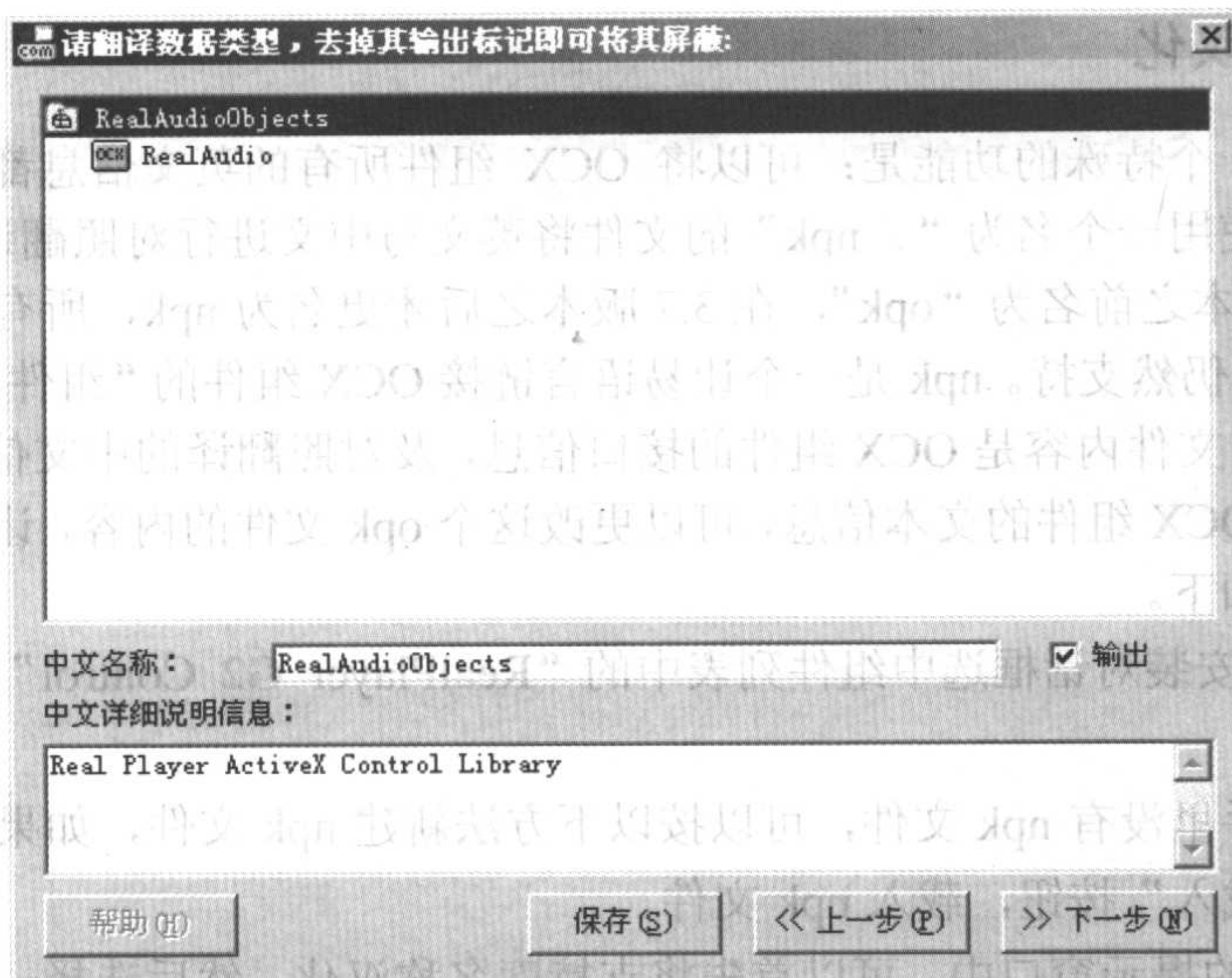


图 16-5 保存注册信息

组件注册信息文件默认后缀为“.npk”。

也可以直接安装本书随书光盘中提供的“rmoc3260.npk”文件。返回上一步，单击“安装”按钮，选中 npk 文件，按照提示完成安装。或者直接将“rmoc3260.npk”文件复制到易语言安装目录的“lib”文件中。

4. 重新启动易语言后，可以在组件面板中看到新添加的 OCX 组件按钮。如图 16-6 所示。



图 16-6 新添加的 OCX 组件

由于各种原因，多次安装同一组件，组件面板会添加多个相同的组件按钮。而用安装工具注销安装后，再次安装，还会出现多个组件按钮。这就需要到易语言安装目录下的“lib”文件夹中寻找注册组件时保存的 npk 文件，手工将其删除。





### 16.1.2 OCX 的汉化

易语言提供一个特殊的功能是：可以将 OCX 组件所有的英文信息都翻译成为中文使用，非常方便。使用一个名为“.npk”的文件将英文与中文进行对照翻译。

npk 在 3.6 版本之前名为“opk”，在 3.7 版本之后才更名为 npk，所有以前的 opk 版本在 3.7 之后的版本仍然支持。npk 是一个让易语言链接 OCX 组件的“组件包装支持库”，是一个纯文本文件，文件内容是 OCX 组件的接口信息，及对照翻译的中文信息，易语言用这个文件控制显示 OCX 组件的文本信息，可以更改这个 opk 文件的内容，让易语言显示中文内容。具体方法如下。

1. 打开组件安装对话框选中组件列表中的“RealPlayer G2 Control”组件，点击“下一步”按钮。

**注意：**如果手里没有 npk 文件，可以按以下方法新建 npk 文件，如果有 npk 文件需要修改，可点击“载入”按钮，载入 npk 文件。

2. 在接下来的提示窗口中，可以首先将支持库名称汉化，然后选择一个欲翻译的数据类型，单击“下一步”按钮继续。如图 16-7、图 16-8、图 16-9 所示。

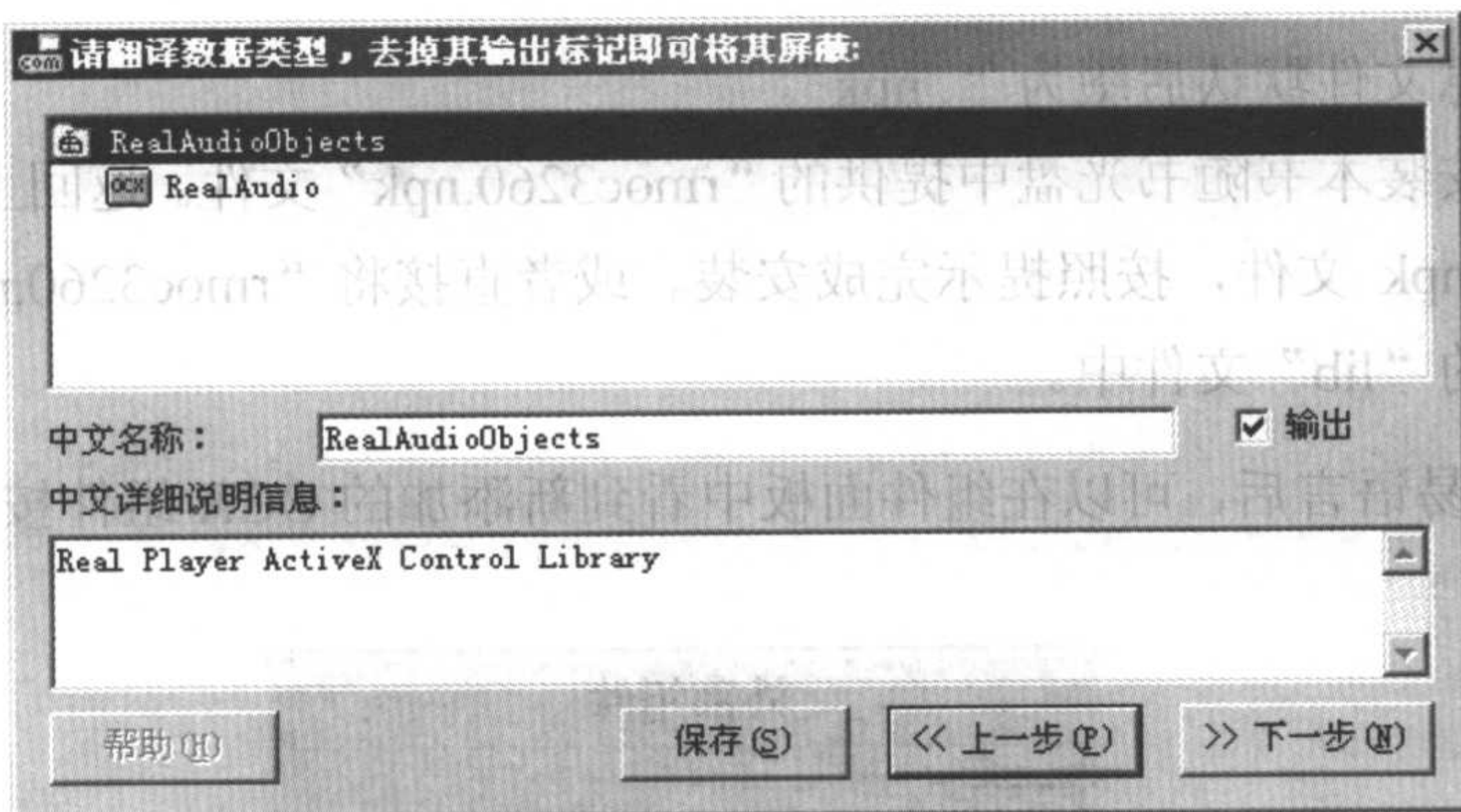


图 16-7 选择支持库名称进行汉化

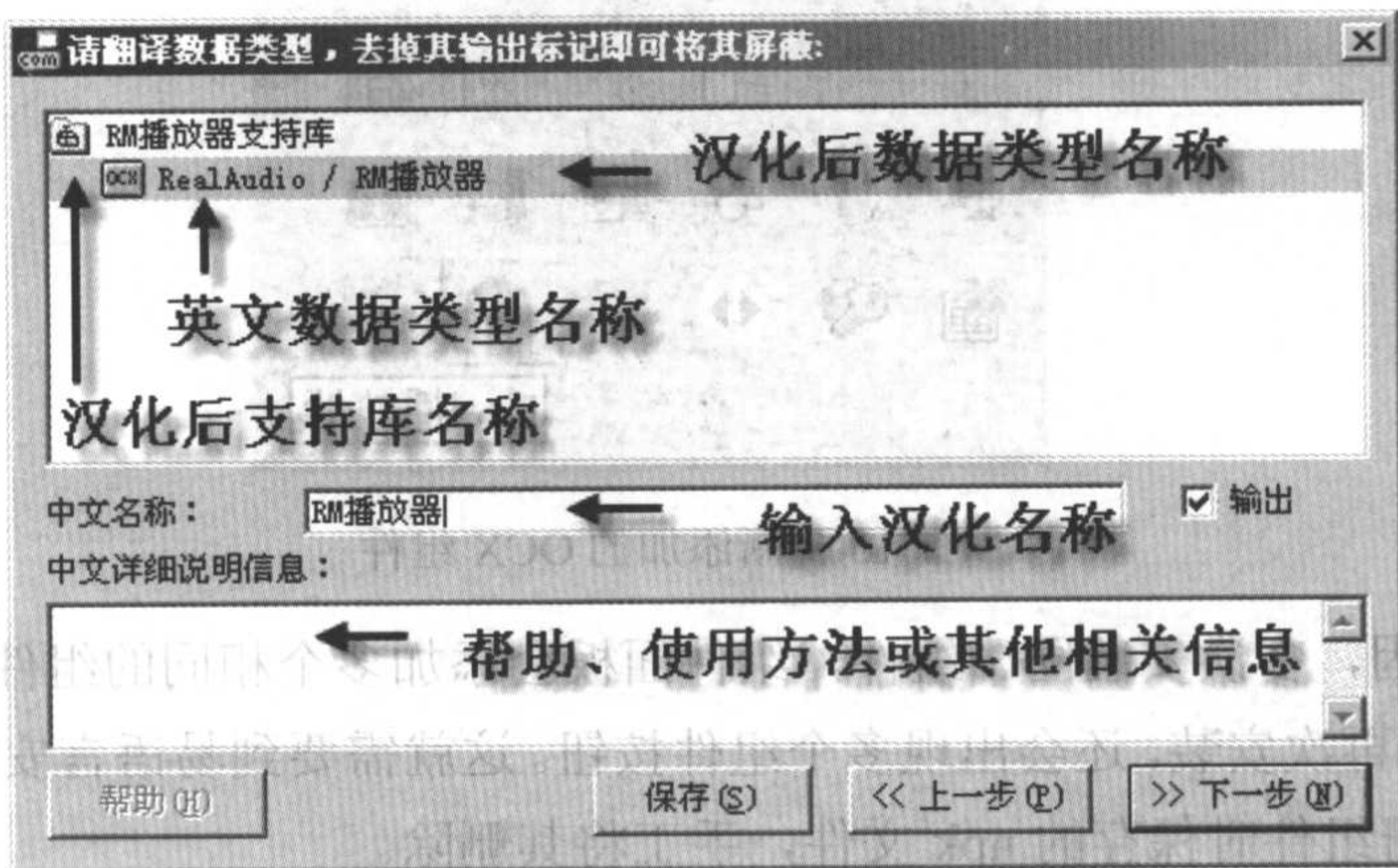


图 16-8 汉化工具窗口



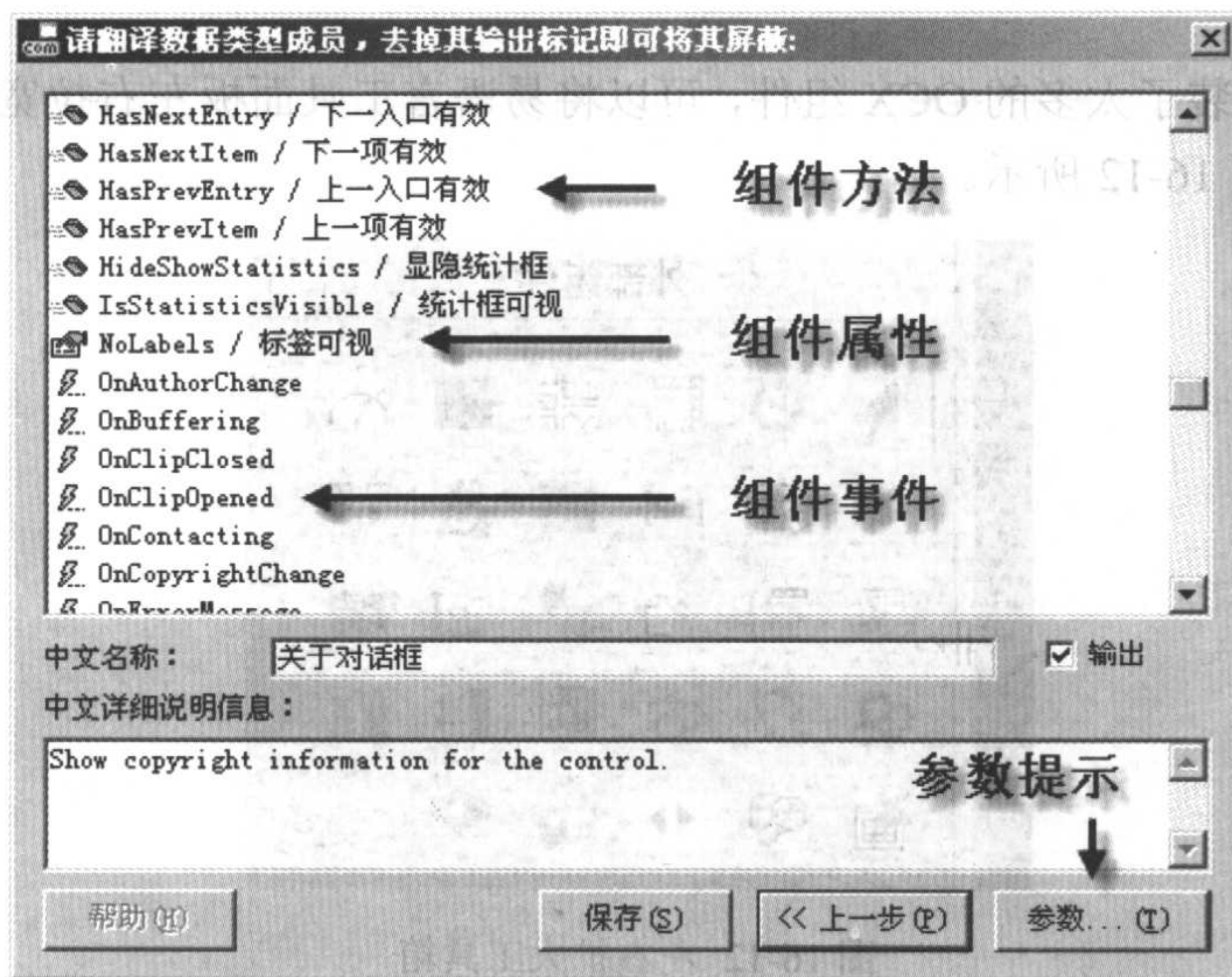


图 16-9 逐条汉化组件的属性、方法和事件名称

3. 单击“保存”按钮，将文件保存为 npk 文件。
4. 在弹出的信息框中提示 npk 文件已创建成功，并询问是否将此文件复制到易语言系统支持库目录中，点击“是”按钮。如图 16-10 所示。

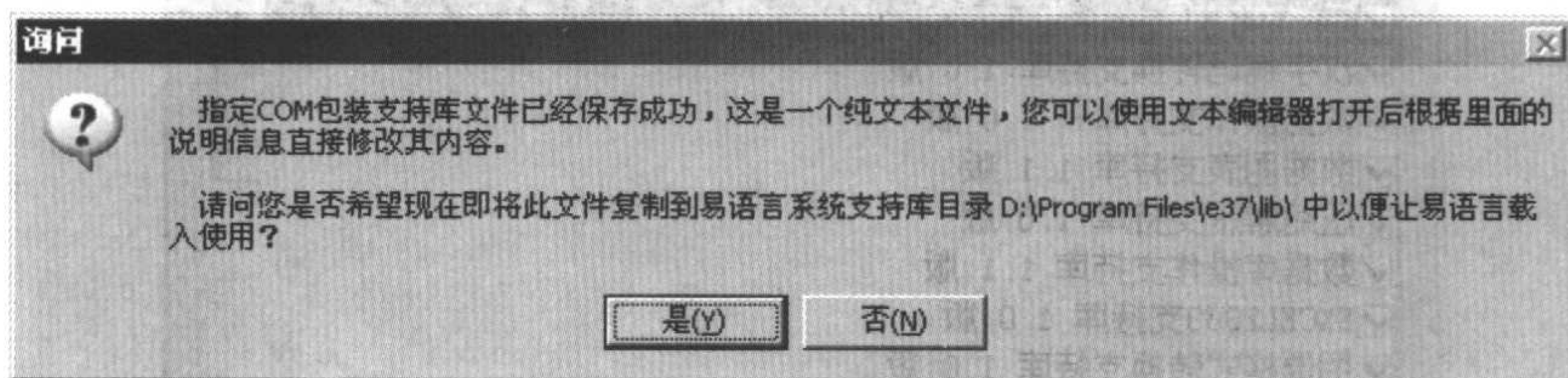


图 16-10 提示创建 npk 文件成功

5. 此时会弹出一个信息框，提示在安装新的支持库之前要先退出程序，询问是否关闭易语言系统，选择“是”。如图 16-11 所示。

建议大家关闭易语言，因为易语言运行时，此支持库文件一直被使用中，从而会使新的文件安装失败。

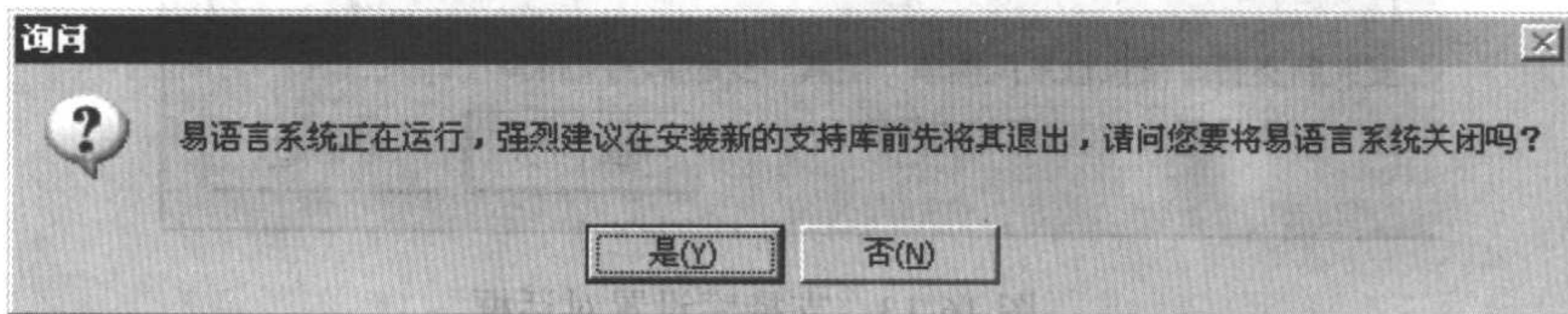


图 16-11 信息框询问是否要关闭易语言系统

6. 如果添加成功，会有信息框显示已经添加完成。  
这样 npk 文件就生成了。重新启动易语言，会在工具面板中看到这个新的 OCX 组件按





钮了。

注意：如果安装了太多的 OCX 组件，可以将易语言工具面板左右拉宽一些，这样会显示其他组件。如图 16-12 所示。



图 16-12 左右扩大工具箱

也可以通过易语言的菜单命令“程序”→“支持库配置”，从而打开支持库设置对话框。如图 16-13 所示。可以将暂时不用的 OCX 组件前面的勾选去掉，这样在控制面板中就不会出现这个 OCX 组件了。

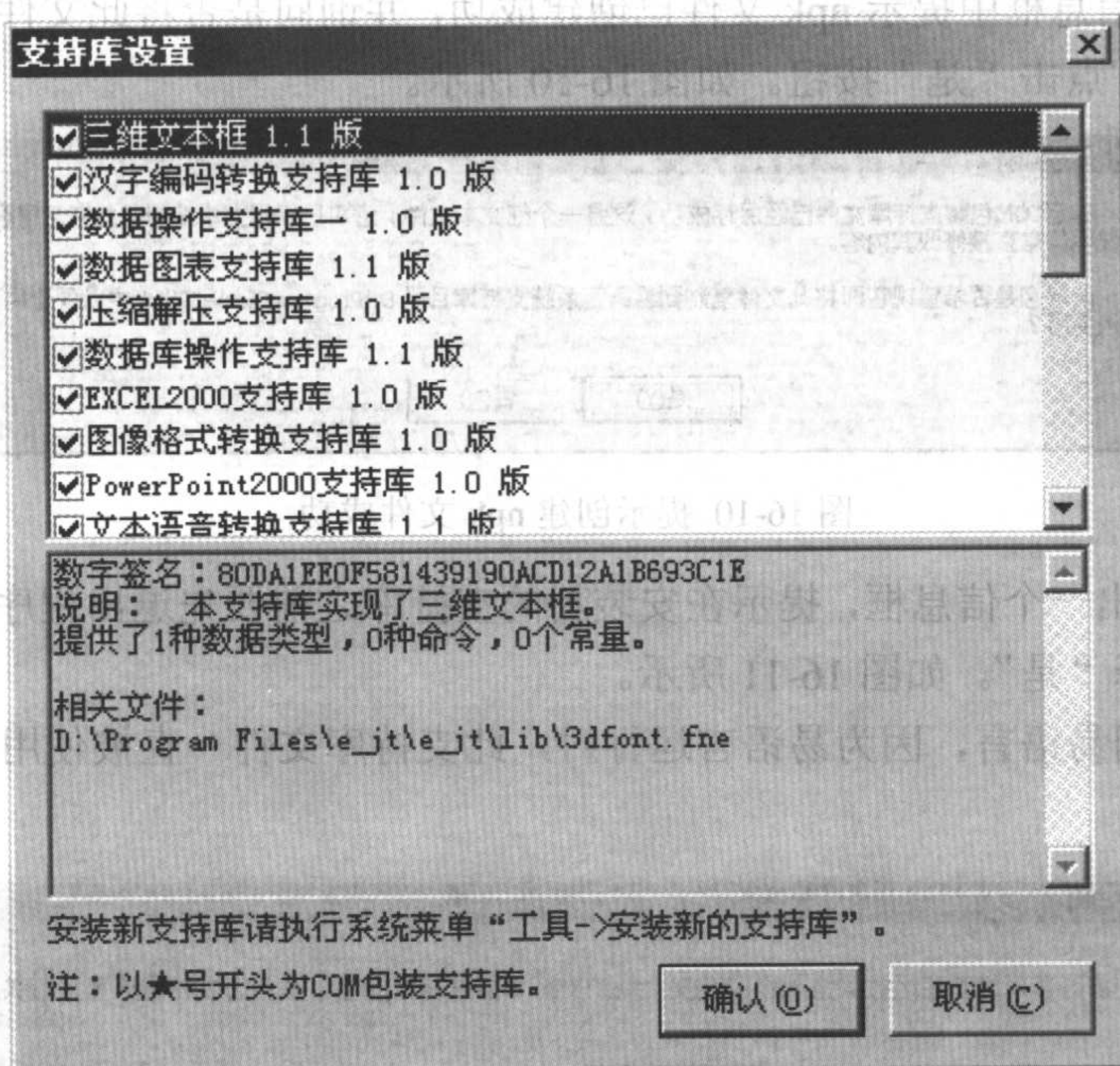


图 16-13 支持库设置对话框

## 16.1.3 OCX 的使用方法

在 OCX 组件安装与汉化完成之后，就可以对其进行使用了，下面就一起来看看几个 OCX 组件的使用例程。



### 例程一：

用 RealPlayer G2 Control (rmoc3260.dll) 组件来制作支持 .rm 等文件格式的播放器，通过点击一个标签组件，打开一个通用对话框，选择一个音乐文件进行播放。

按上面所讲述的步骤安装与汉化 RealPlayer G2 Control (rmoc3260.dll) 组件。

先来看一看 RealPlayer G2 Control (rmoc3260.dll) 组件的一些常用属性：

#### 1. “可视部分”属性

RM 组件一共有 3 种状态，每一种状态对应一种控制界面。其中一种是播放控制，就是前进，后退那些，第二种是播放，就是纯粹的播放屏幕（大多数人需要的是这一种）。第三种是状态条。3 种状态是通过一个属性控制的，一个空间每次只能显示为一种状态。

#### 2. “窗口名”属性

文本值。返回或设置这个组件的窗口名。

#### 3. “自动跳转”属性

逻辑值。返回/设置组件是否会自动地激活超链接事件，与音频流结合。如果“真”，跳转超链接事件会交替闪光。

#### 4. “自动播放”属性

逻辑值。返回/设置一旦某个数据源可用，组件是否会自动地开始播放。

#### 5. “组件名”属性

文本值。返回/设置组件名，组件以相同的名字工作。

#### 6. “音频源”属性

返回/设置音频夹来源。

在系统支持库面板中也可以找到这个组件的方法。如图 16-14 所示。

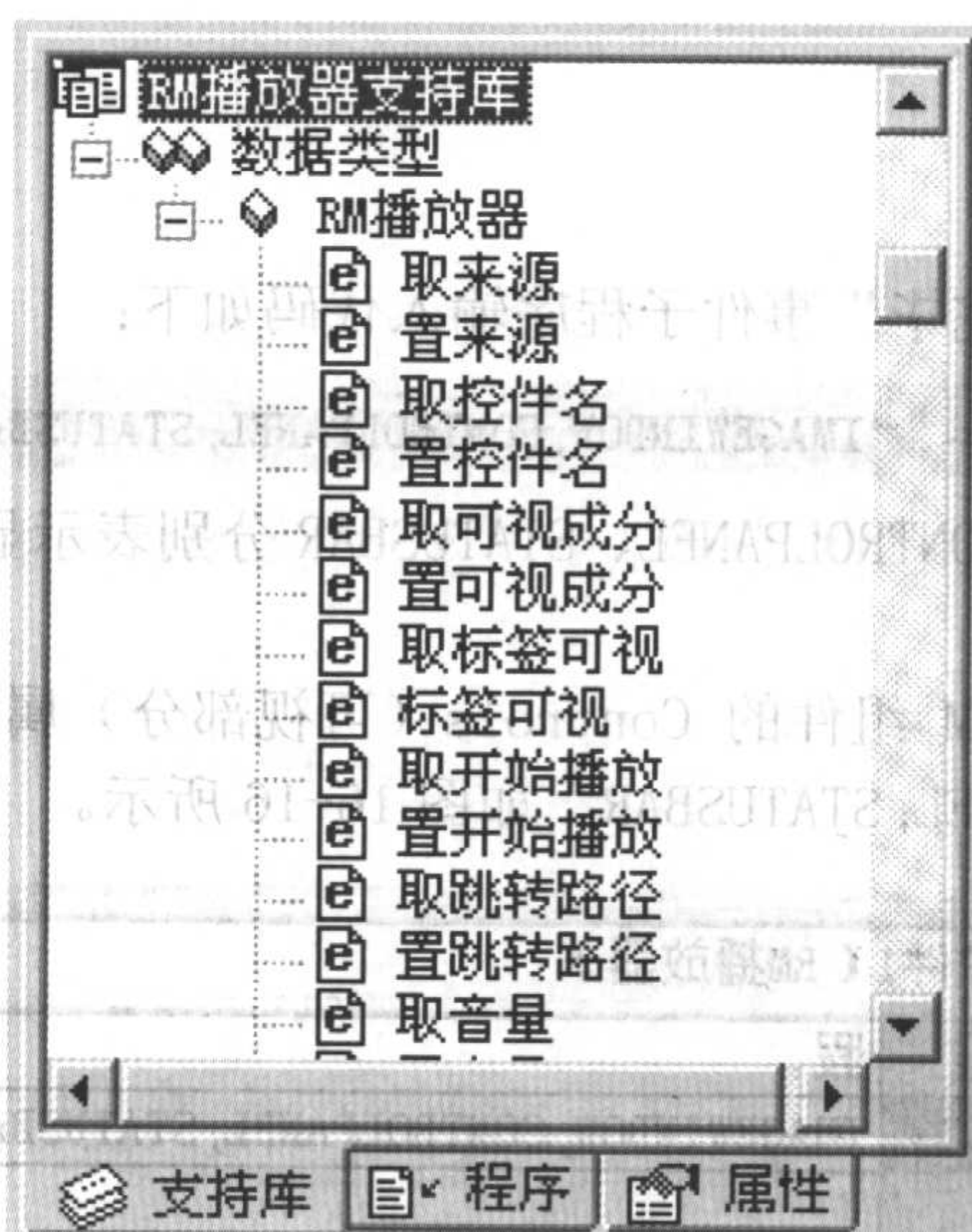


图 16-14 RM 组件的部分方法

### 具体操作步骤：

1. 新建一个“Windows 窗口程序”，在启动窗口中添加一个“RM 播放器”组件、一





个通用对话框组件、一个标签组件。如图 16-15 所示。

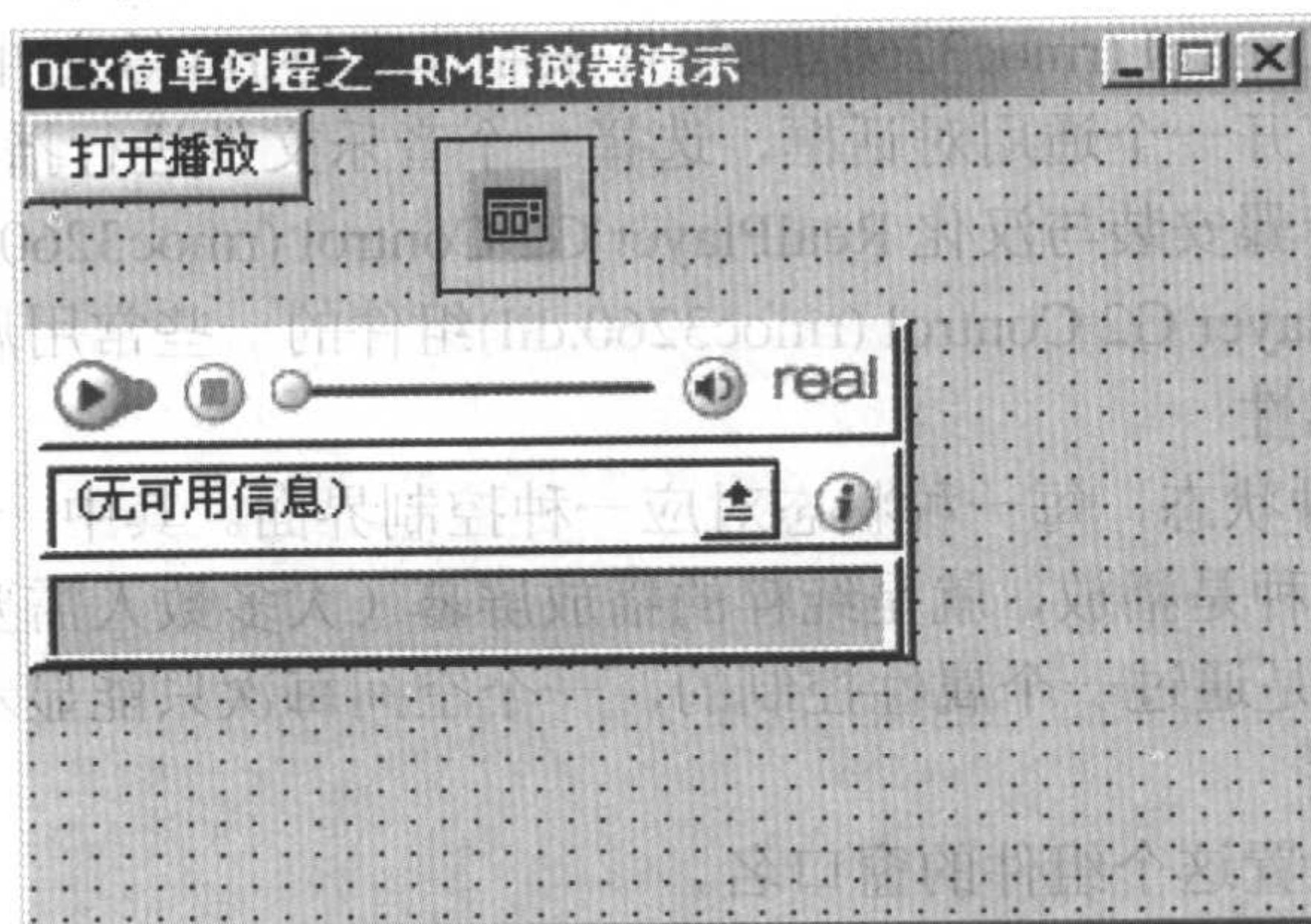


图 16-15 程序设计界面组件排列

## 2. 分别设置三个组件的属性:

### “RM 播放器”组件:

窗口名: 播放器;

自动播放: 真;

标签可视: 真;

### “通用对话框”组件:

类型: 打开文件;

过滤器: “\*.rm 文件|\*.rm|\*.mp3 文件|\*.mp3|\*.wav 文件|\*.wav|\*.dat 文件|\*.dat|所有文件|\*.rm;\*.mp3;\*.wav;\*.mp2;\*.dat”

### “标签”组件:

标题: 打开播放;

边框: 浅凹入式

## 3. “\_\_启动窗口\_创建完毕”事件子程序输入代码如下:

```
Real播放件1.可视部分 = “IMAGEWINDOW, CONTROL PANEL, STATUSBAR”
```

其中: IMAGEWINDOW、CONTROL PANEL、STATUSBAR 分别表示显示视频播放窗口、控制条和状态条。

除此之外, 也可以在 RM 组件的 Controls (可视部分) 属性中添加如下文本型代码: “IMAGEWINDOW, CONTROL PANEL, STATUSBAR” 如图 16-16 所示。



图 16-16 属性表中设置“可视部分”属性



设置后，可以看到组件外观已经改变成了视频播放窗口形状。如图 16-17 所示。可以试着减去此属性中的一项，看看组件外观的显示效果。



16-17 设置组件外观

4. “\_标签 1\_鼠标左键被按下”事件子程序中，输入代码如下：

```

--- 如果真 (通用对话框1.打开 ()
    Real播放件1.控件名 = 通用对话框1.文件名
    Real播放件1.音频源 = 通用对话框1.文件名
    Real播放件1.自动播放 = 真

```

5. 为了使窗口关闭时停止播放音乐，可以在“\_启动窗口\_将被销毁”事件子程序中输入代码如下：

```

Real播放件1.停止播放 ()

```

6. 一个简单的 RM 播放软件做好了。试运行，选择后缀为“RM”的视频文件，就可以播放。如图 16-18 所示。

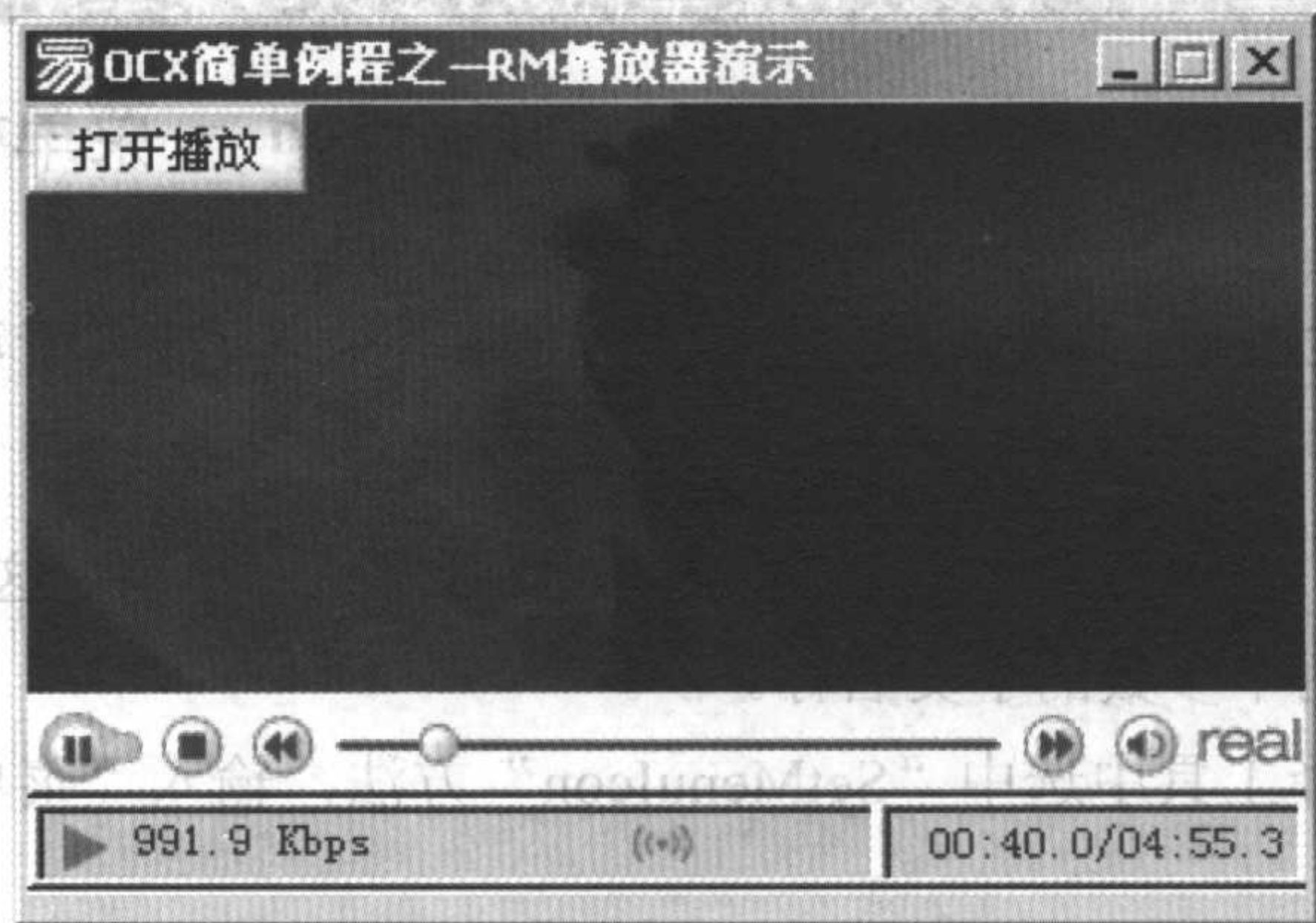


图 16-18 播放 RM 视频文件





## 例程二:

1. 注册随书光盘所提供的“EMenu30.ocx”组件文件。
2. 由于例程“菜单组件 NPK.e”是在易语言 3.7 之前编写的，必须安装随书光盘中的“EMenu30.npk”才可以打开程序，而且组件的方法、属性全部为中文。运行程序，通过子菜单项“菜单背景”选择背景图片。效果如图 16-19 所示。

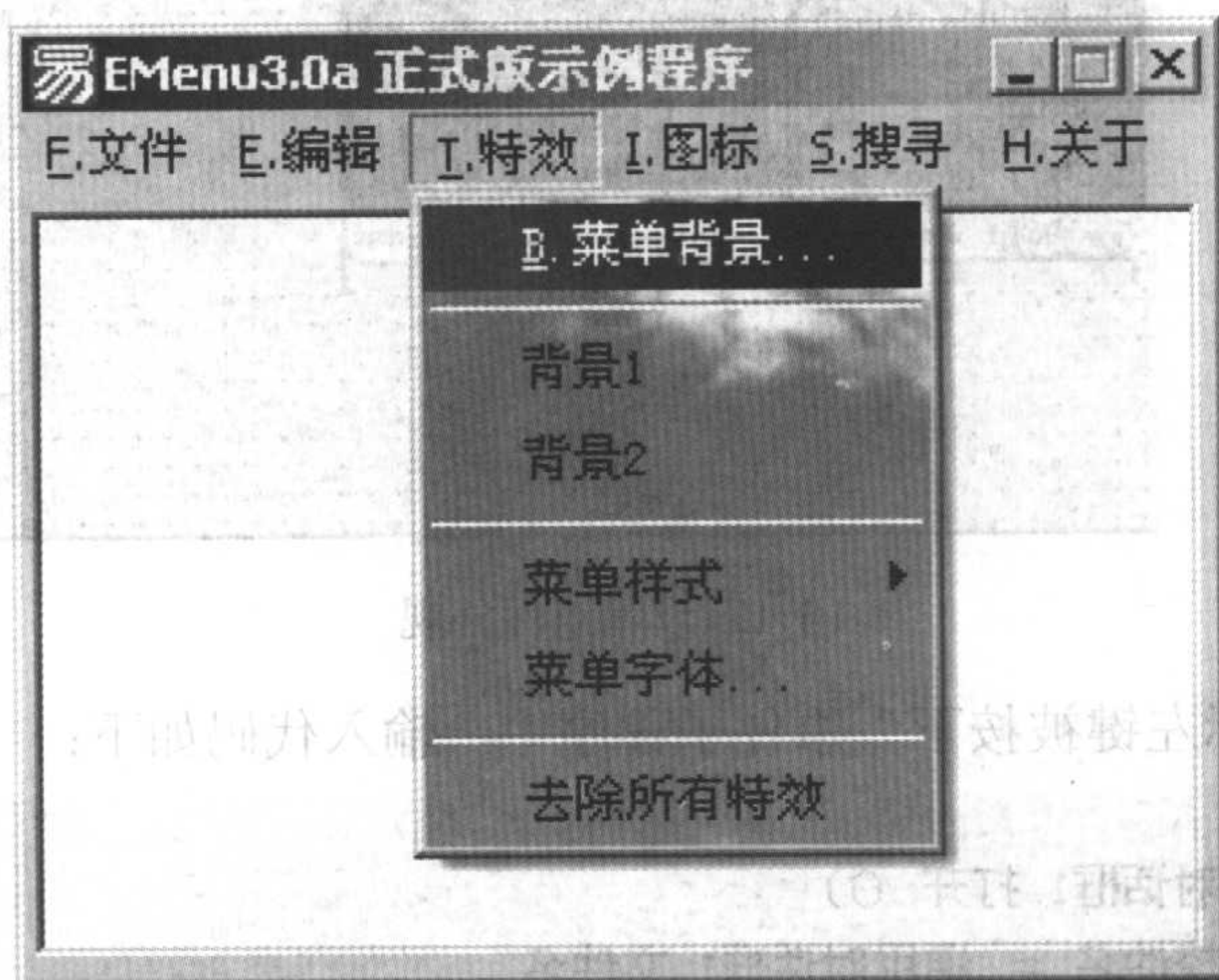


图 16-19 设置菜单背景图片

3. 易语言 3.7 中，用“EMenu30.ocx”组件直接安装，生成组件的方法、属性不再是中文，使用时很不方便。下面用原来的 npk 文件，将组件汉化。

在 npk 文件中按方法、属性、常量的顺序保存。

使用“封装类型库及 OCX 组件”工具，保存 npk 文件之前将组件汉化。如 opk 文件中方法下的“SetMenuIcon”方法，如下所示。

```
##1 EMenu.SetMenuIcon = 菜单图标  
设置菜单图标 (参数1:菜单标题,参数2:菜单图标索引)
```

```
##1 _EMenu.SetMenuIcon.MenuCaption = 菜单标题
```

```
##1 _EMenu.SetMenuIcon.IconIndex = 图标索引
```

其中“菜单图标”是等号前方法名称的中文名称，下一行是对这个方法的中文解释。下面两行是此方法的两个参数的中文名称。

这样就可以在翻译工具中选中“SetMenuIcon”方法，输入“菜单图标”。如图 16-20 所示。



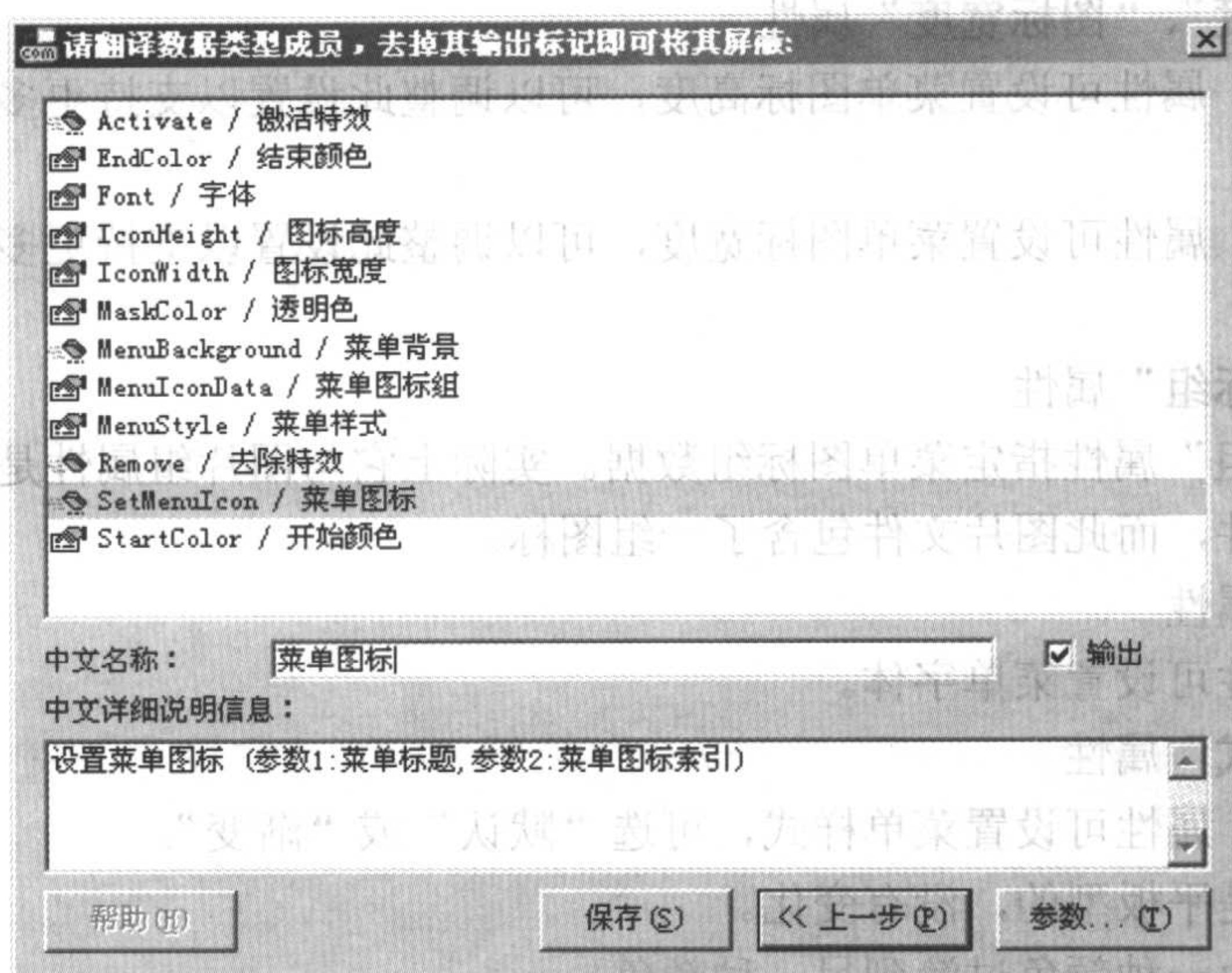


图 16-20 翻译组件

标题为“参数...”按钮被激活，提示此方法有参数。单击按钮或双击方法进入参数翻译窗口，分别将“菜单标题”、“图标索引”输入。最后保存为 npk 文件。组件安装后组件命令被添加到支持库面板。如图 16-21 所示。

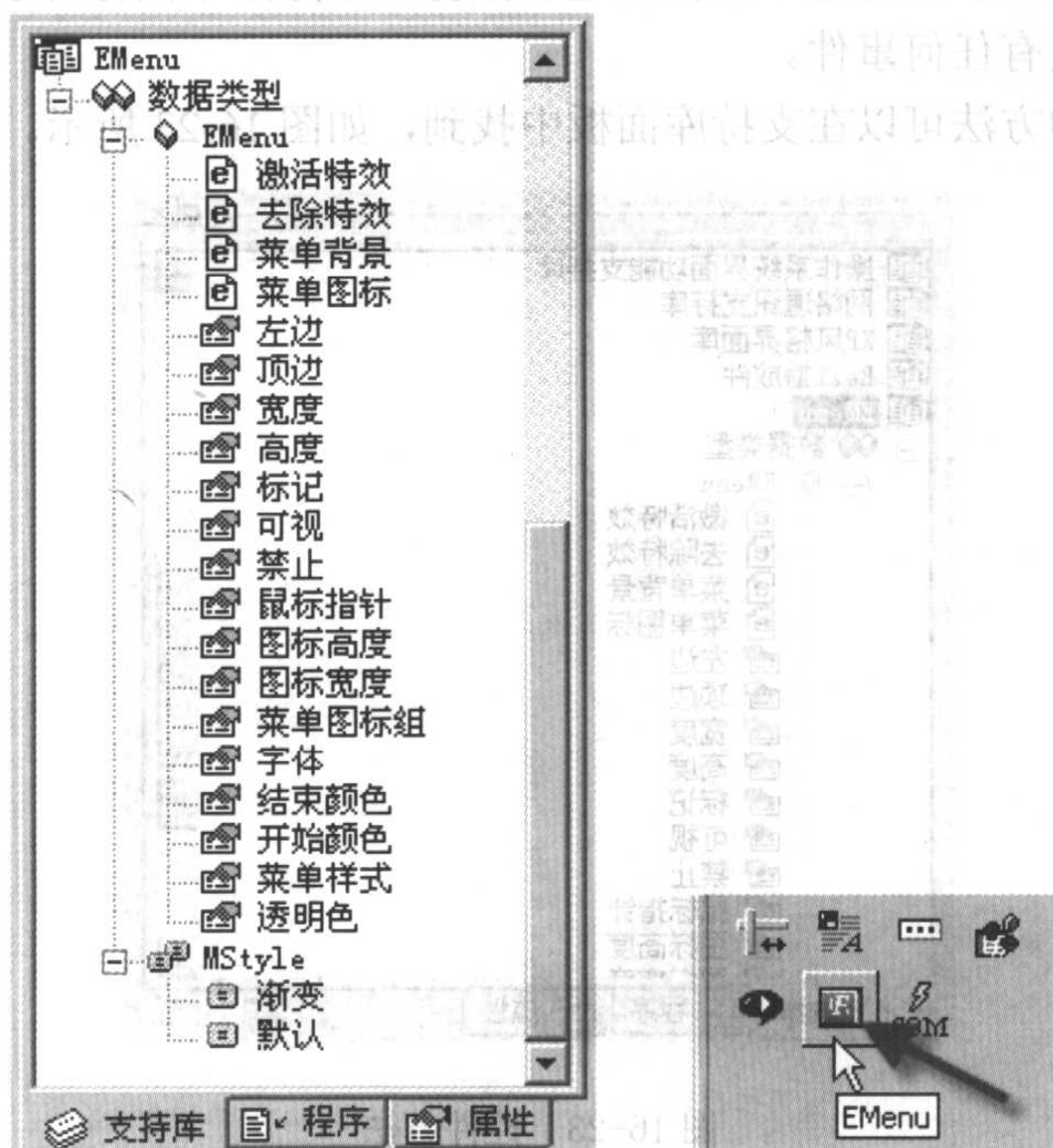


图 16-21 组件支持库图

16-22 新增的控件

最后，可以在工具面板中找到这个新的控件。如图 16-22 所示。

后面将称“EMenu”组件为“菜单组件”，下面来看它的一些常用属性。





## 1. “图标高度”、“图标宽度”属性

“图标高度”属性可设置菜单图标高度，可以调整此设置以支持更多的图片组数据类型。

“图标宽度”属性可设置菜单图标宽度，可以调整此设置以支持更多的图片组数据类型。

## 2. “菜单图标组”属性

“菜单图标组”属性指定菜单图标组数据。实际上它与图片组属性是一样的，也可以输入一个图片文件，而此图片文件包含了一组图标。

## 3. “字体”属性

“字体”属性可设置菜单字体。

## 4. “菜单样式”属性

“菜单样式”属性可设置菜单样式，可选“默认”或“渐变”。

“默认”即是平板型的，没有变化。

“渐变”即由一种颜色过渡到另一种颜色。

## 5. “开始颜色”、“结束颜色”属性

当“菜单样式”属性为“渐变”时，“开始颜色”属性设置渐变开始颜色值。

当“菜单样式”属性为“渐变”时，“结束颜色”属性设置渐变结束颜色值。

## 6. “透明色”属性

“透明色”属性用于指定菜单图标组透明颜色，去除菜单图标外层“黑边”。

菜单按钮控件没有任何事件。

菜单按钮控件的方法可以在支持库面板中找到，如图 16-23 所示。

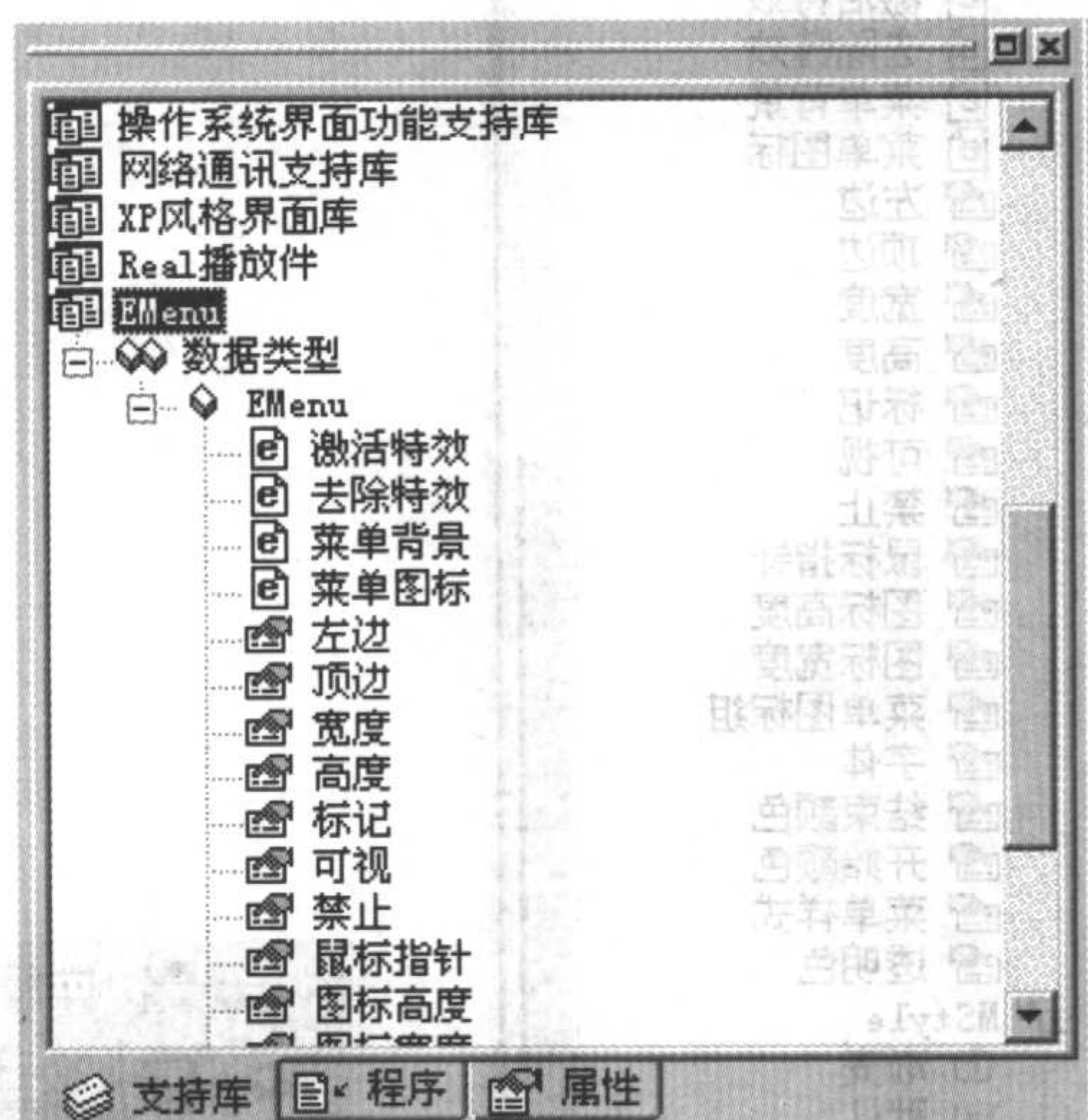


图 16-23 控件方法

下面，看一下此例程中用到的该控件方法：

## 1. “激活特效( )”方法

“激活特效( )”方法可激活按钮菜单特效，只有激活特效，才可以使用这个控件的功



能。实际上这个方法是固定的，即在启动窗口创建完毕时都必须使用这个方法命令以激活菜单特效。

## 2. “菜单图标 ( )” 方法

“菜单图标 ( )” 方法设置菜单图标。

参数 1：菜单标题。

参数 2：菜单图标索引。

例如：

菜单组件. 菜单图标 (渐变. 标题, 1)

## 3. “去除特效 ( )” 方法

“去除特效 ( )” 方法去除菜单特效。

注意：去除后无法再次调用“激活特效 ( )”开启菜单特效。

## 4. “菜单背景 ( )” 方法

“菜单背景 ( )” 方法指定菜单背景。

注意：不要载入空数据！否则程序会出错。

具体操作步骤：

1. 新建一个“Windows 窗口程序”，将图片存在图片资源中。如图 16-24 所示。

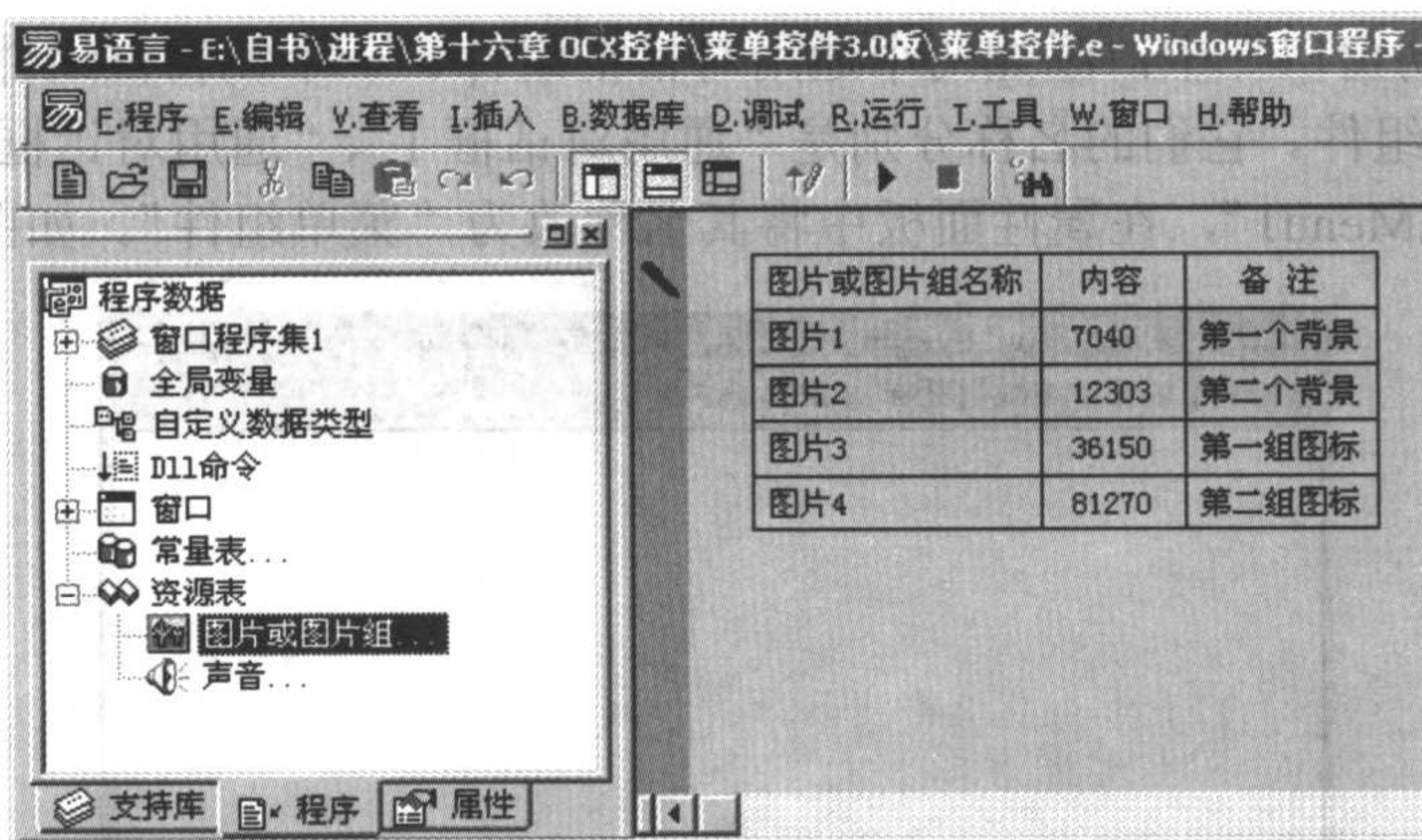


图 16-24 图片资源

2. 回到“\_启动窗口”，右键单击窗口，选择“菜单编辑器”选项。如图 16-25 所示。

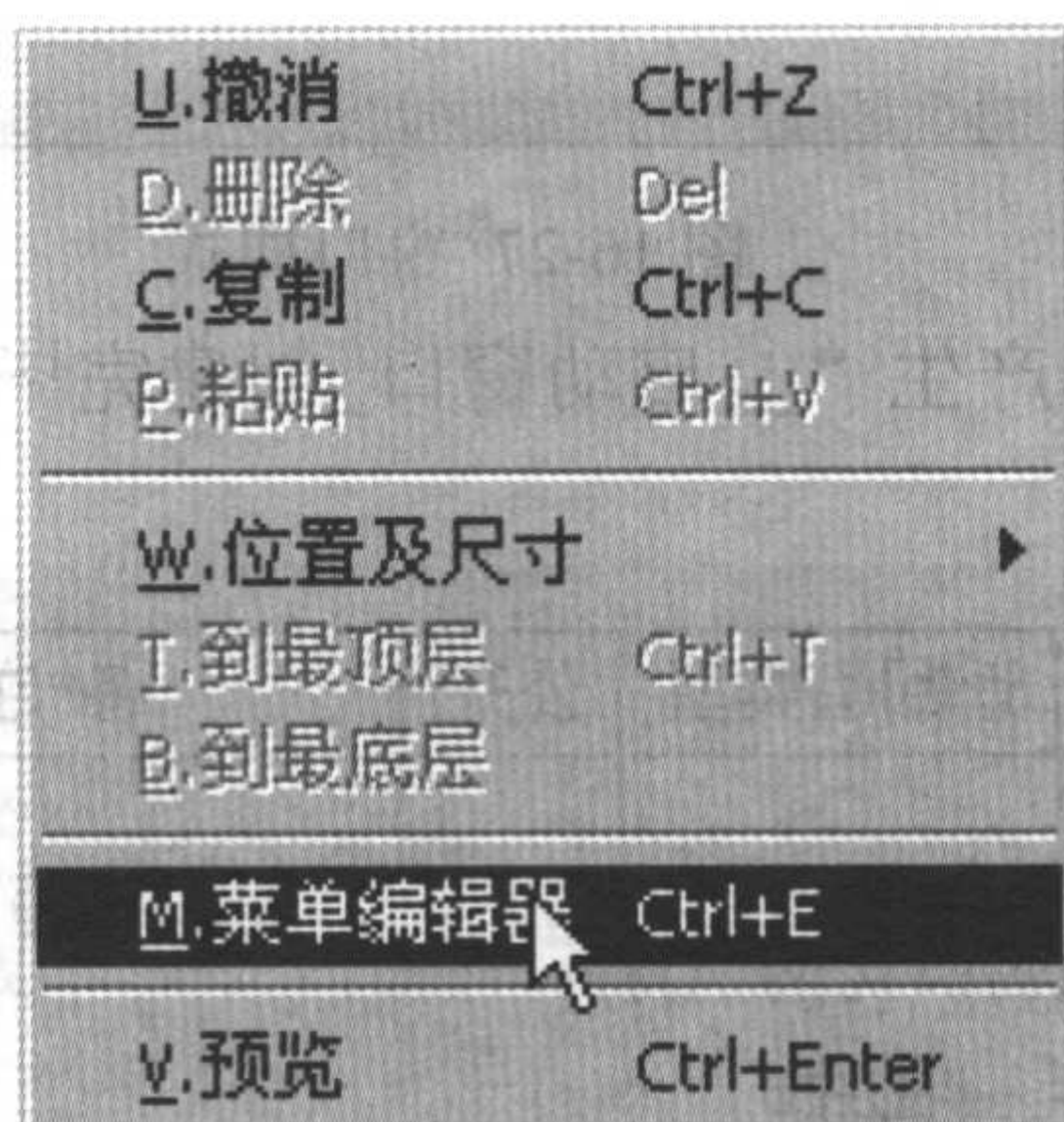


图 16-25 选择“菜单编辑器”选项





在弹出的“菜单编辑器”窗口中添加该程序所需要的菜单名称。如图 16-26 所示：

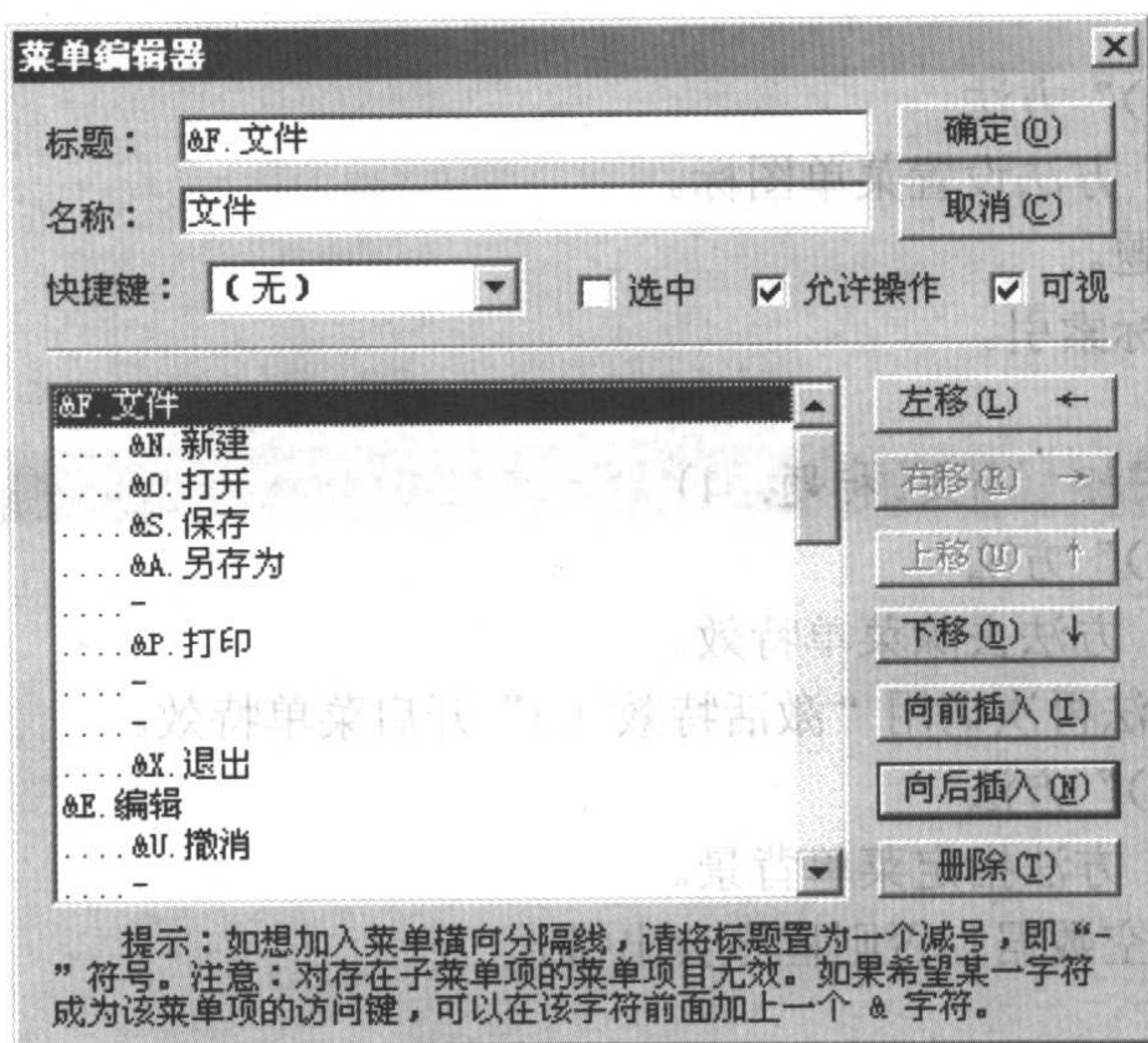


图 16-26 菜单编辑器

3. 在“\_启动窗口”中，添加两个通用对话框组件、一个编辑框组件（用来修饰窗口之用）、一个菜单组件。它们的名称分别是“通用对话框 1”、“通用对话框 2”，菜单组件刚添加后名称是“EMenu1”，在属性面板中将其名称改为“菜单组件”。如图 16-27 所示。

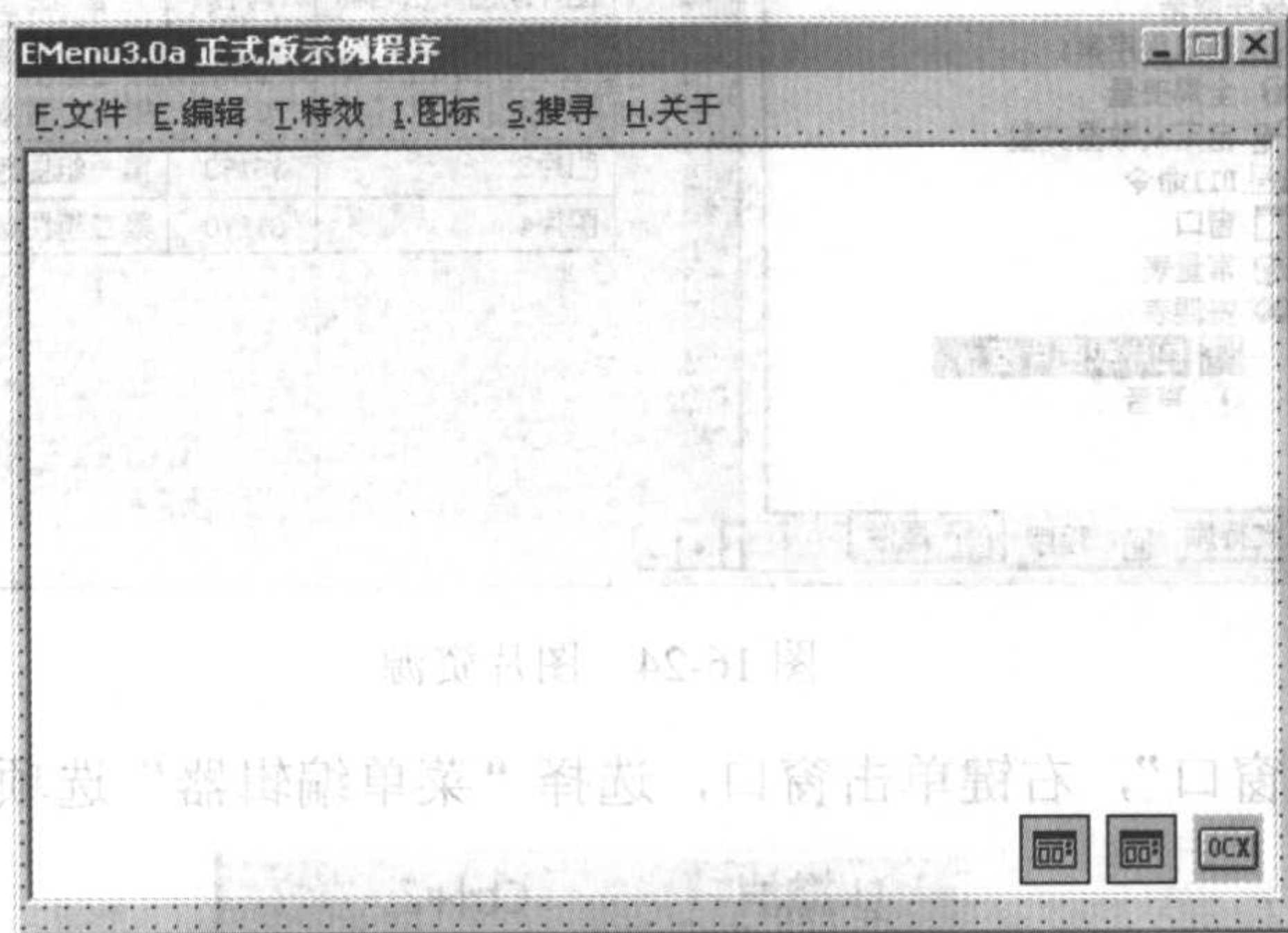


图 16-27 窗口样式

4. 双击“\_启动窗口”，产生“\_\_启动窗口\_创建完毕”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

菜单组件. 激活特效 (取窗口句柄 0)

菜单组件. 菜单图标组 = #图片3

设置菜单图标 0



其中“设置菜单图标 ()”是一个自定义的子程序，通过调用它来添加菜单中的图标：

子程序名	返回值类型	公开	备注
设置菜单图标			

菜单组件. 菜单图标 (新建. 标题, 11)  
 菜单组件. 菜单图标 (打开. 标题, 12)  
 菜单组件. 菜单图标 (保存. 标题, 13)  
 菜单组件. 菜单图标 (打印. 标题, 19)  
 菜单组件. 菜单图标 (撤消. 标题, 8)  
 菜单组件. 菜单图标 (剪切. 标题, 5)  
 菜单组件. 菜单图标 (复制. 标题, 6)  
 菜单组件. 菜单图标 (粘贴. 标题, 7)  
 菜单组件. 菜单图标 (日期. 标题, 26)  
 菜单组件. 菜单图标 (查找. 标题, 17)  
 菜单组件. 菜单图标 (标准. 标题, 1)

4. 设置“通用对话框 2”的类型为“字体选择”；设置“通用对话框 1”的类型为“打开文件”。设置“EMenu1”的菜单样式为“渐变”。

5. 在“\_启动窗口”中选择“文件”菜单中的“退出”选择，产生“\_退出\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_退出_被选择			

销毁 0

6. 在“\_启动窗口”中选择“特效”菜单中的“菜单背景”选择，产生“\_菜单背景\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_菜单背景_被选择			

变量名	类型	静态	数组	备注
图片对象	对象			

```
通用对话框1. 文件名 = ""
通用对话框1. 过滤器 = "图片文件[*.*gif;*.bmp]|*.gif;*.bmp"
--- 如果真 (通用对话框1. 打开 ()
    图片对象. 创建图片对象 (读入文件 (通用对话框1. 文件名))
    菜单组件. 菜单背景 (图片对象)
    指定菜单背景后，请不要尝试清除菜单背景！（会使菜单特效强行被清除）
```

7. 在“\_启动窗口”中选择“特效”菜单中的“背景 1”选项，产生“\_背景 1\_被选择”事件子程序，在其中输入以下代码：





子程序名	返回值类型	公开	备注
_背景1_被选择			

变量名	类型	静态	数组	备注
图片对象	对象			

图片对象.创建图片对象 (#图片1)

菜单组件.菜单背景 (图片对象)

指定菜单背景后, 请不要尝试清除菜单背景! (会使菜单特效强行被清除)

--- 如果真 (背景2.选中)

背景2.选中 = 假

菜单组件.菜单图标 (背景2.标题, -1) ' "-1" 清除菜单图标

背景1.选中 = 真

菜单组件.菜单图标 (背景1.标题, 1)

8. 在“\_启动窗口”中选择“特效”菜单中的“背景2”选项, 产生“\_背景2\_被选择”事件子程序, 在其中输入以下代码:

子程序名	返回值类型	公开	备注
_背景2_被选择			

变量名	类型	静态	数组	备注
图片对象	对象			

图片对象.创建图片对象 (#图片2)

菜单组件.菜单背景 (图片对象)

指定菜单背景后, 请不要尝试清除菜单背景! (会使菜单特效强行被清除)

--- 如果真 (背景1.选中)

背景1.选中 = 假

菜单组件.菜单图标 (背景1.标题, -1)

背景2.选中 = 真

菜单组件.菜单图标 (背景2.标题, 1)

9. 在“\_启动窗口”中选择“特效”菜单中“菜单样式”中的“标准”选项, 产生“\_标准\_被选择”事件子程序, 在其中输入以下代码:

子程序名	返回值类型	公开	备注
_标准_被选择			

菜单组件.菜单样式 = 0

--- 如果真 (渐变.选中)

渐变.选中 = 假

菜单组件.菜单图标 (渐变.标题, -1)

标准.选中 = 真

菜单组件.菜单图标 (标准.标题, 1)

10. 在“\_启动窗口”中选择“特效”菜单中“菜单样式”中的“渐变”选项, 产生“\_渐变\_被选择”事件子程序, 在其中输入以下代码:



子程序名	返回值类型	公开	备注
_渐变_被选择			

```

菜单组件.菜单样式 = 1
--- 如果真 (标准.选中)
    标准.选中 = 假
    菜单组件.菜单图标 (标准.标题, -1)
    渐变.选中 = 真
    菜单组件.菜单图标 (渐变.标题, 1)

```

11. 在“\_启动窗口”中选择“特效”菜单中的“菜单字体”选项，产生“\_菜单字体\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_菜单字体_被选择			

变量名	类型	静态	数组	备注
字体变量	字体			

```

--- 如果真 (通用对话框2.打开 ())
    字体变量.加粗 = 通用对话框2.加粗
    字体变量.倾斜 = 通用对话框2.倾斜
    字体变量.删除线 = 通用对话框2.删除线
    字体变量.下划线 = 通用对话框2.下划线
    字体变量.字体名称 = 通用对话框2.字体名称
    字体变量.字体大小 = 通用对话框2.字体大小
    菜单组件.字体 = 字体变量

```

12. 在“\_启动窗口”中选择“特效”菜单中的“去除所有特效”选项，产生“\_去除所有特效\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_去除所有特效_被选择			

```

    如果真 (信息框 (“确定要去除所有特效吗？” + #换行符 + #换行符 +
    “注意：清除后，只有重新启动程序才能再次激活特效！”， #询问图标 +
    #是否钮, ) = #是钮)
    菜单组件.去除特效 ()

```

13. 在“\_启动窗口”中选择“图标”菜单中的“16×16”选项，产生“\_尺寸 16\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_尺寸16_被选择			

```

菜单组件.图标宽度 = 16
菜单组件.图标高度 = 16
菜单组件.菜单图标组 = #图片3

```





14. 在“\_启动窗口”中选择“图标”菜单中的“24×24”选项，产生“\_尺寸 24\_被选择”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备注
_尺寸24_被选择			

菜单组件.图标宽度 = 24

菜单组件.图标高度 = 24

菜单组件.菜单图标组 = #图片4

15. 在“\_启动窗口”中选择编辑框 1，在该组件的事件列表中选择“鼠标右键被放开”事件，产生“\_编辑框 1\_鼠标右键被放开”事件子程序，在其中输入以下代码：

子程序名	返回值类型	公开	备 注		
_编辑框1_鼠标右键被放开	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

弹出菜单 (特效, , )

返回 (假)

16. 试运行，看一下菜单效果，如图 16-28 所示。

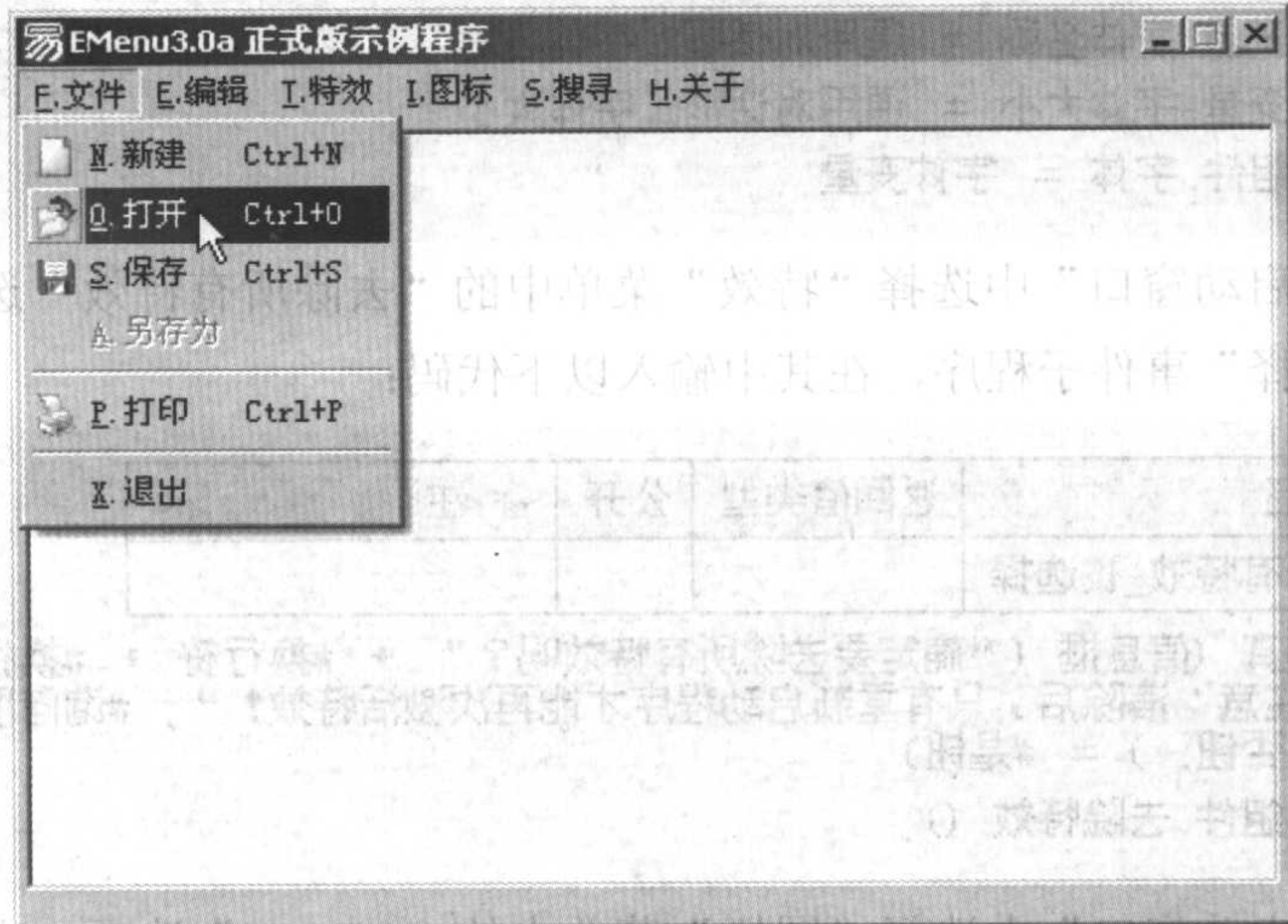


图 16-28 试运行后的效果

## 16.2 类型库的封装和使用

类型库是一个描述信息的集合，这些描述信息涉及类、接口、接口方法、接口方法的



参数，以及相关的常量和事件。类型库通过 ITYPELIB 接口来访问，对于易语言、Delphi、VB 等智能开发环境，无法使用 C++ 头文件来找到接口的方法和属性信息，这时可以把类型库当作独立于编程语言的数据类型定义。

类型库通常位于 \*.tlb, \*.olb, \*.dll, \*.ocx 等文件中。

易语言提供了专门的工具用于封装、汉化类型库。

### 16.2.1 类型库的封装

易语言可以读取类型库中的信息，并允许将其中的英文翻译为中文。下面一步步的来做。

首先请选择菜单“工具”→“封装类型库和 OCX 组件”。您的易语言“工具”菜单可能与此不同，因为当您安装第三方支持库时，它通常会在“工具”菜单的底部添加自己的菜单项。

**注意：**也可以直接运行易语言安装目录下的 \tools\packcom.exe 程序以启动“封装类型库和 OCX 组件”。

之后，易语言将自动搜索当前已在操作系统中注册的类型库以及 OCX 组件，然后显示出所有列表，如图 16-29 所示。

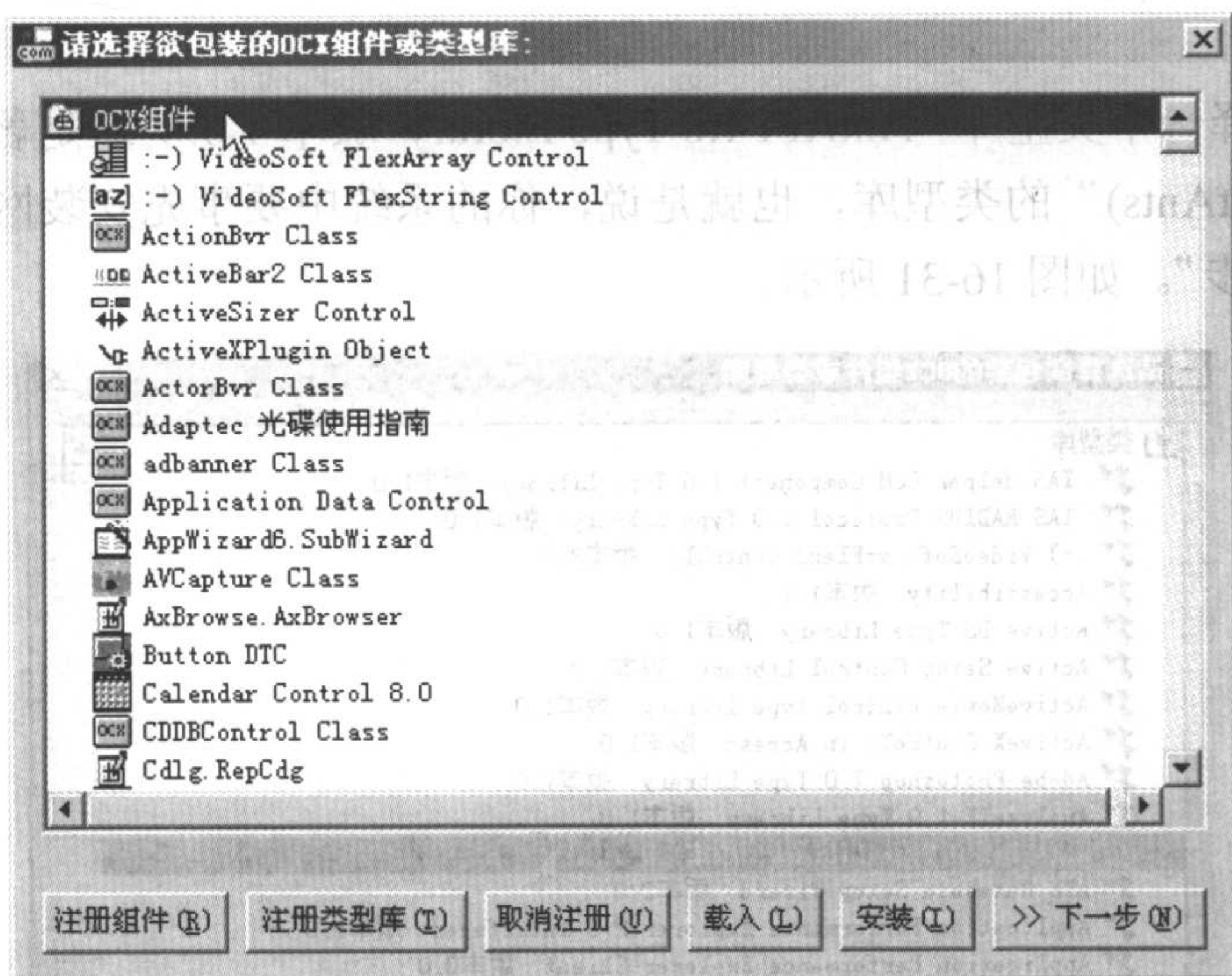


图 16-29 已注册的 OCX 组件和类型库（一）

整个列表分为两大类：“OCX 组件”和“类型库”。两个大类都已展开，而“OCX 组件”在前面，所以看不到“类型库”。下面双击整个列表的第一行“OCX 组件”，使所有 OCX 组件折叠起来，就看到了“类型库”，如图 16-30 所示。



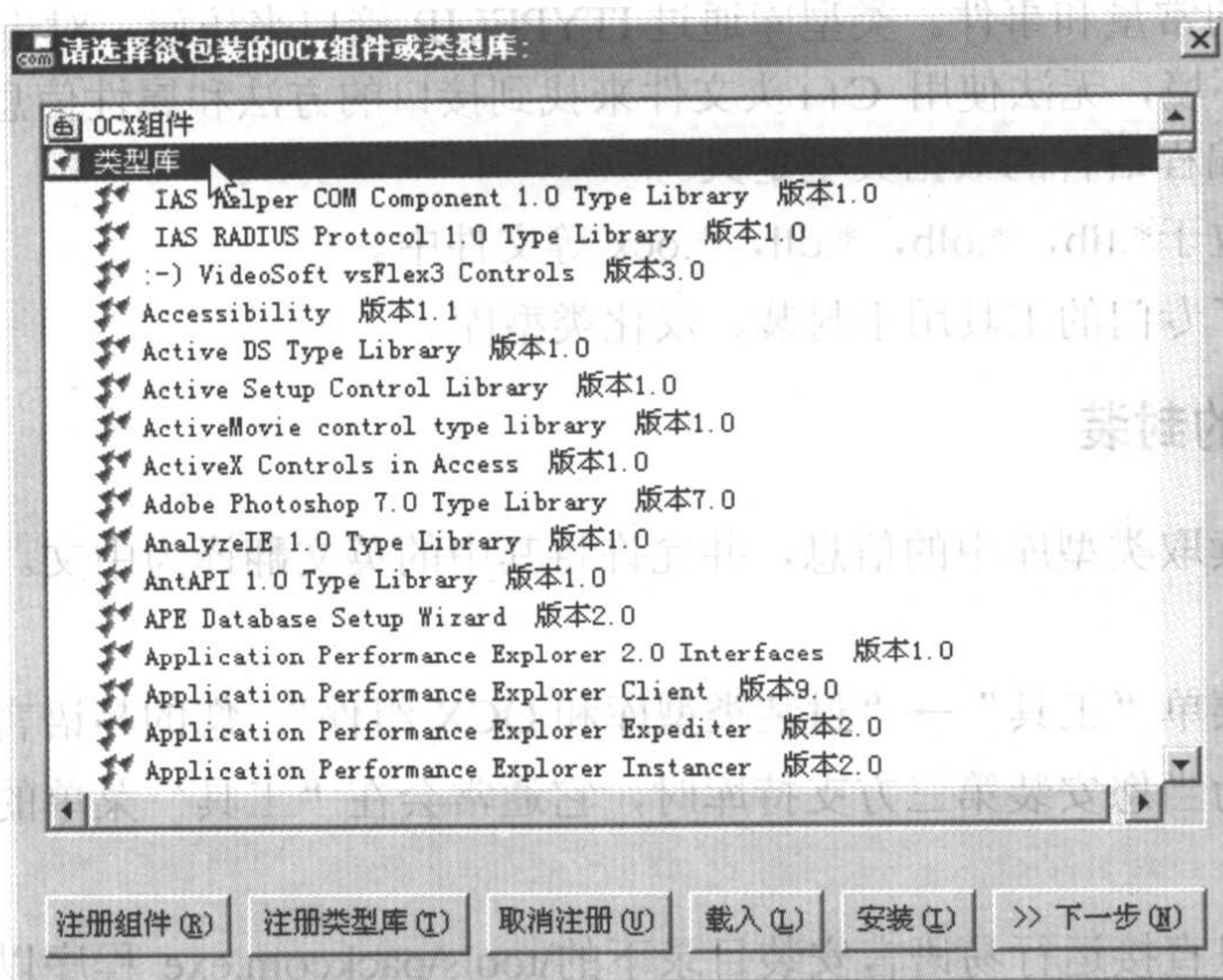


图 16-30 已注册的 OCX 组件和类型库（二）

通过窗口右侧的滚动条，就可以大致判断已注册的类型库数量是多么庞大，总有六七百吧！再加上前面的二三百个 OCX 组件，哇，都可以直接拿来使用。做易语言用户真是幸福！

下面大家选择一个类型库“AntAPI 1.0 Type Library 版本 1.0”，这是著名的网络下载工具“网络蚂蚁(NetAnts)”的类型库，也就是说，你的系统中要事先安装网络蚂蚁。然后单击按钮“>>下一步”。如图 16-31 所示。



图 16-31 选择“AntAPI”类型库，并单击“>>下一步”按钮

下面看到了该类型库中定义的三个数据类型，分别是“AntAPIObj”、“AntCatchObj”和“AntMonObj”（注：最后一个实际上是空类型），如图 16-32 所示。



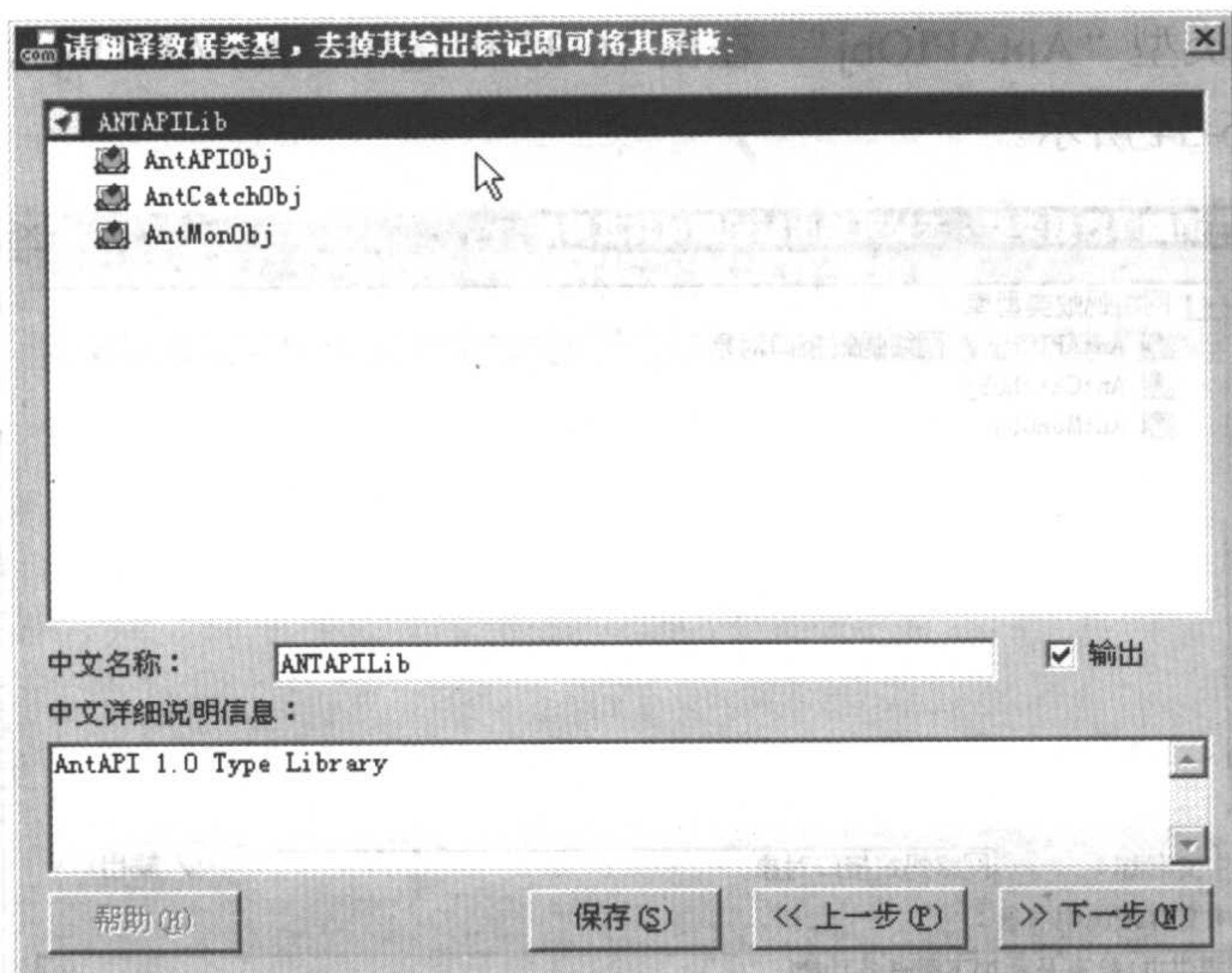


图 16-32 AntAPILib 类型库

**注意：**图 16-32 中的下半部分，他允许客户设置该类型库的中文信息（中文名称、中文详细说明信息），以及是否“输出”（如果不选择“输出”，易语言将不对其进行封装）。

“输出”显然是应该选中的。对于“中文名称”和“中文详细说明信息”，里面默认填写的是英文信息，大家可以选择将其翻译为中文，也可以保持原有英文。一般来说，如果是自己使用，不翻译也是可以的；如果要拿给别人使用，最好还是认真的翻译为中文。因为不见得所有易语言用户的英文水平都很好。翻译之后的好处是：在易语言中使用该类型库时，可以“全中文”编程。如果不打算翻译的话，现在就可以直接单击“保存”按钮了。作为示例，简单地翻译一下，如图 16-33 所示。

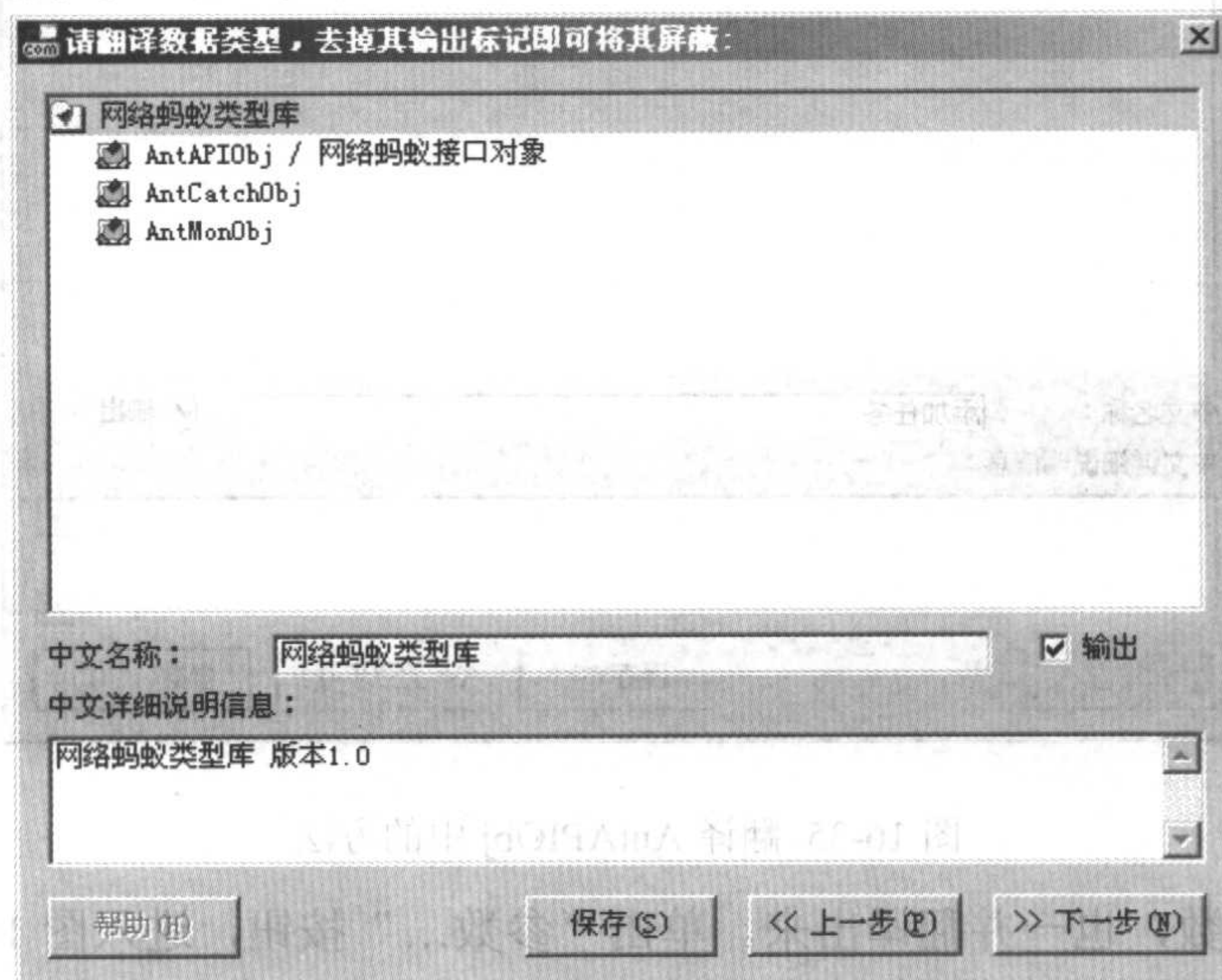


图 16-33 翻译 AntAPILib 类型库

上面只是翻译了类型库的信息，其中的各数据类型，以及各数据类型下的属性、方法、常量，都可以翻译为中文。





下面翻译数据类型“AntAPIObj”。在图 16-33 中选中“AntAPIObj”，填写窗口底部的中文信息，如图 16-34 所示。

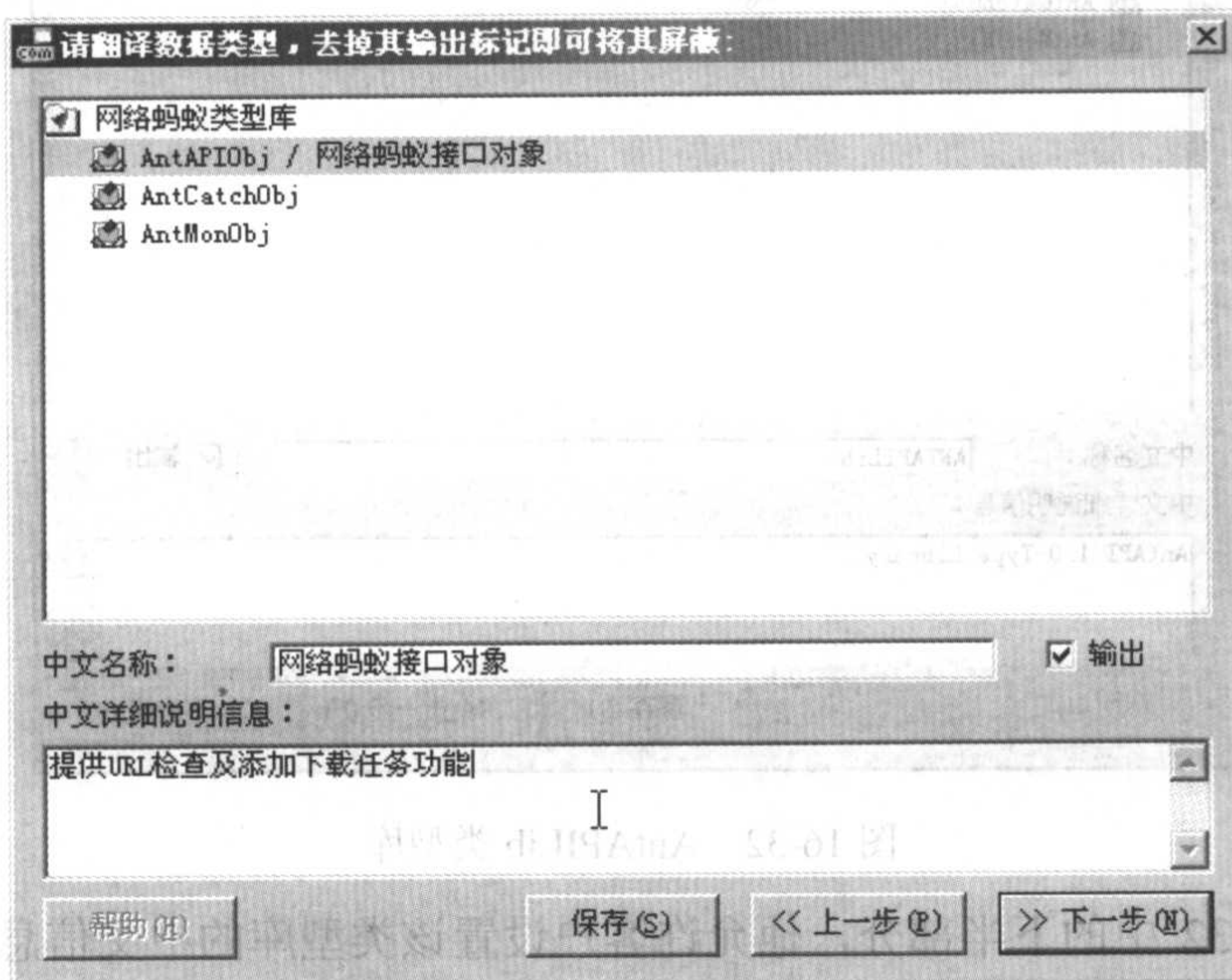


图 16-34 翻译 AntAPIObj 数据类型

接着单击“>>下一步”按钮，翻译 AntAPIObj 下的三个方法，如图 16-35 所示。

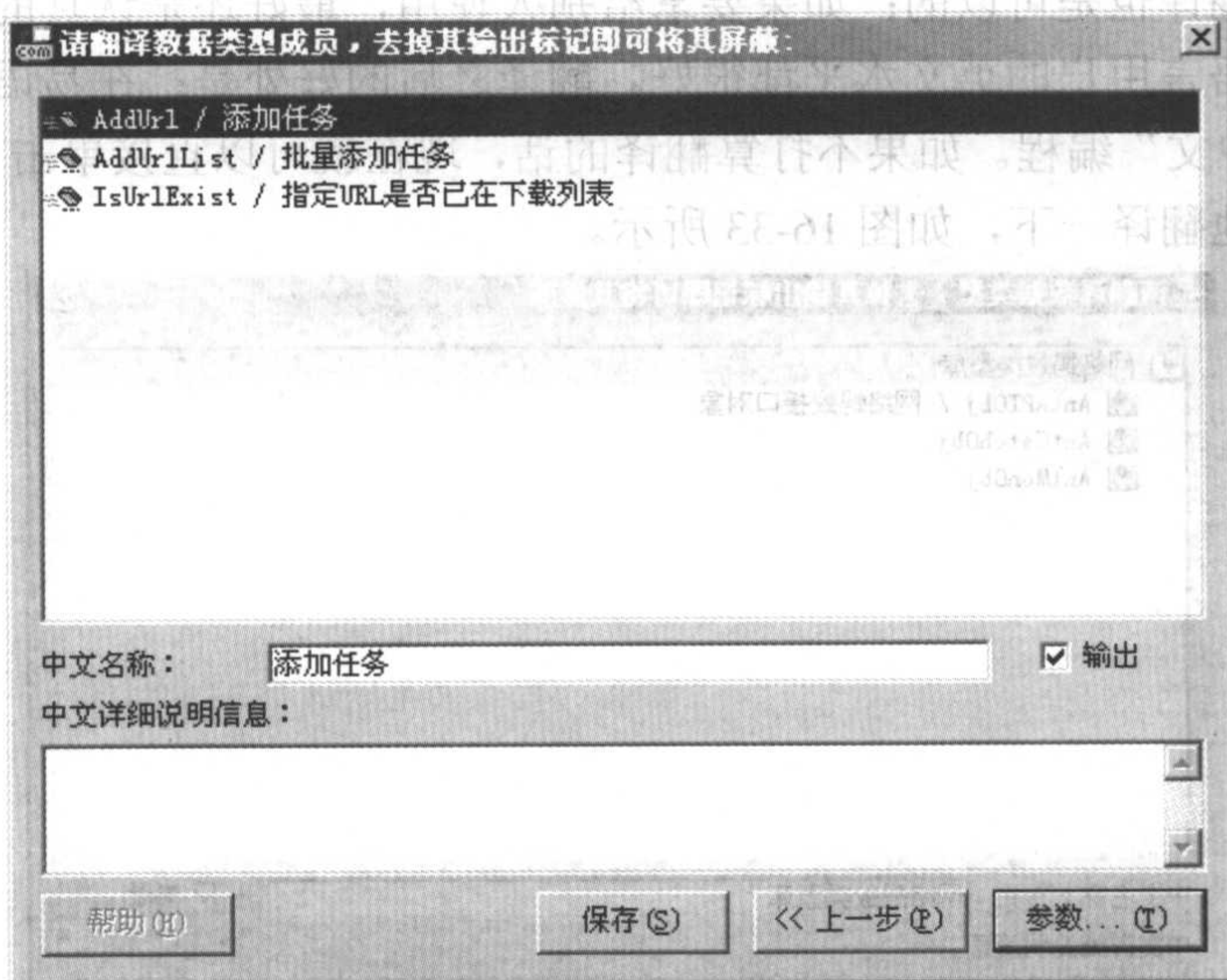


图 16-35 翻译 AntAPIObj 中的方法

各方法都有参数，也一并翻译出来。单击“参数...”按钮，进入图 16-36 所示窗口：



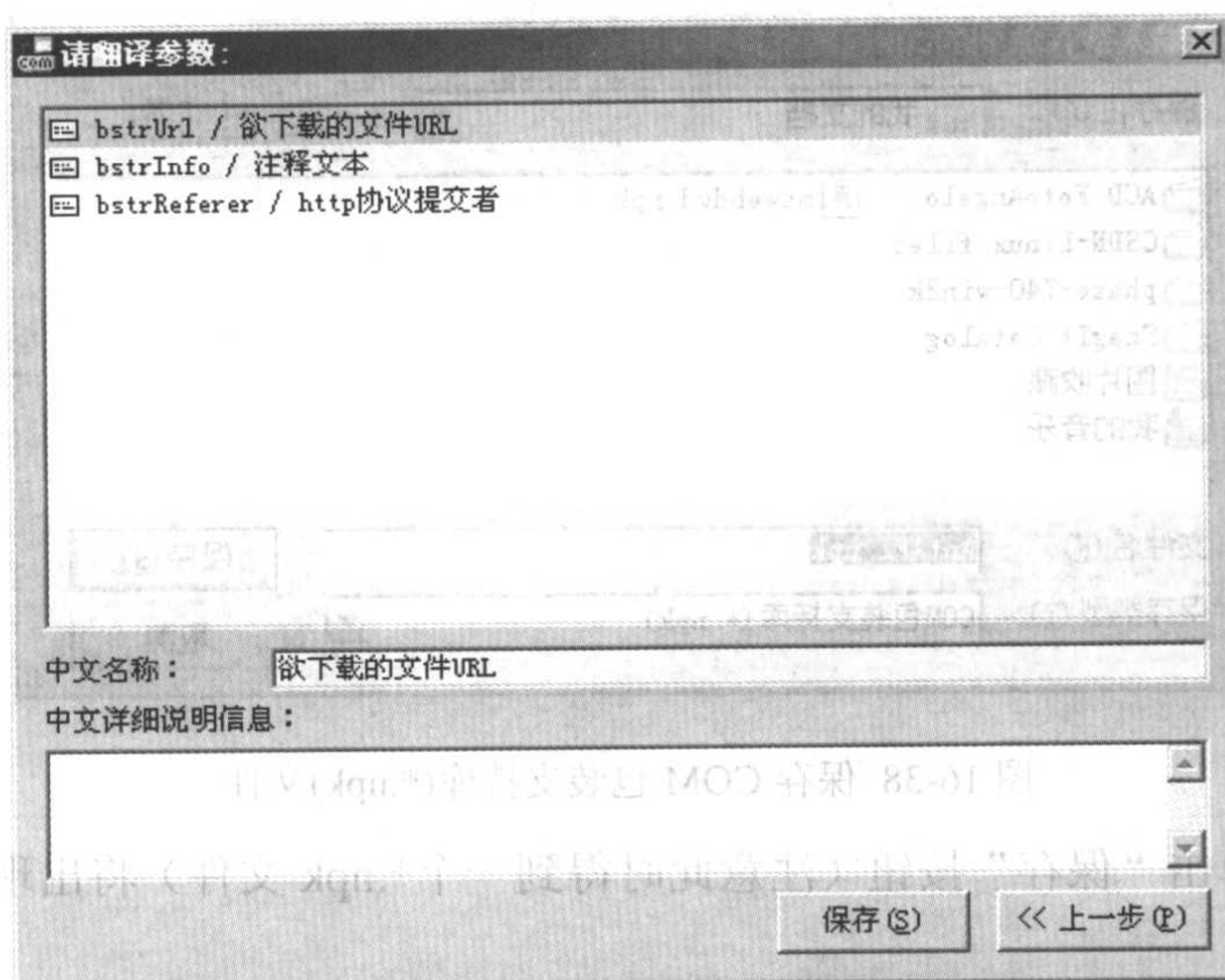


图 16-36 翻译 AddUrl 方法的参数

完成后单击“<<上一步”返回上一窗口。

按照同样的方法，可将所有数据类型及其方法以及方法的参数都翻译出来。此处省略。

另外，因为数据类型 AntMonObj 是空的，易语言中反正也用不着它，可以将它的“输出”标志去掉，如图 16-37 所示。

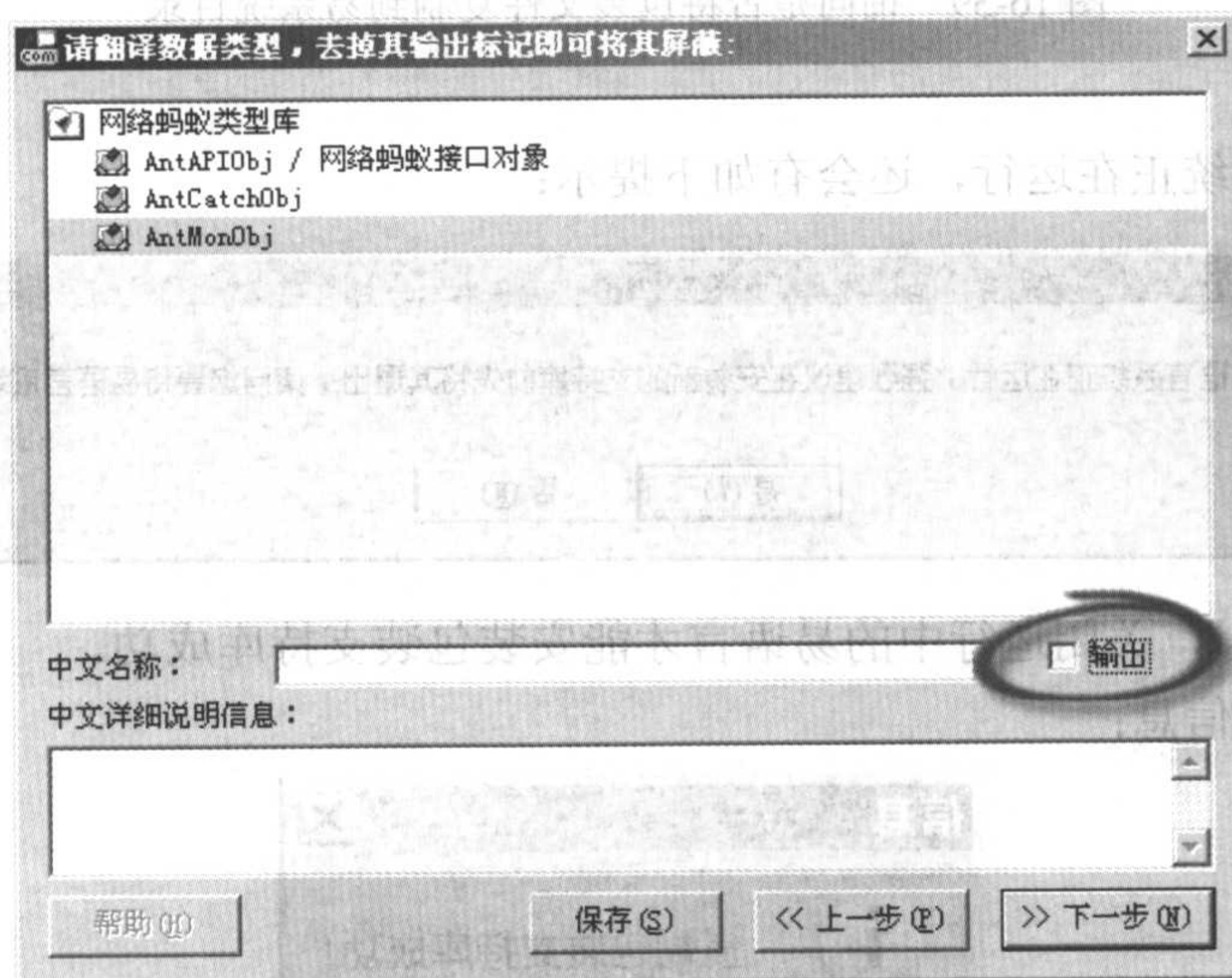


图 16-37 去掉 AntMonObj 的“输出”标记

好了，通过以上操作，汉化工作基本完成了。现在请单击“保存”按钮（如果不需要汉化的话，早在图 16-33 中就可以单击“保存”按钮了）将出现保存文件对话框，如图 16-38 所示。



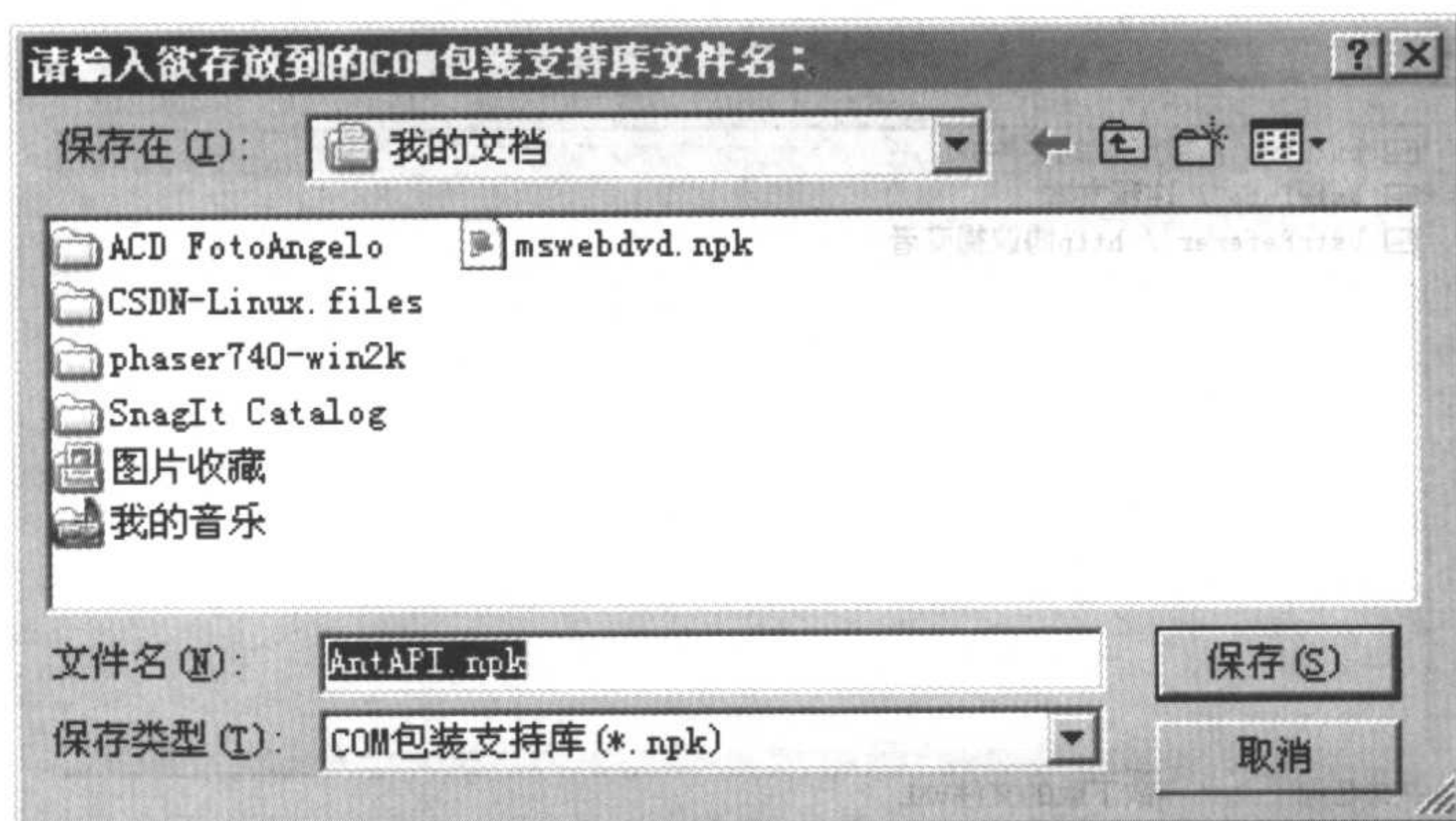


图 16-38 保存 COM 包装支持库(\*.npk)文件

选择路径后单击“保存”按钮（注意此时得到一个\*.npk 文件）将出现如图 16-39 所示的信息框。

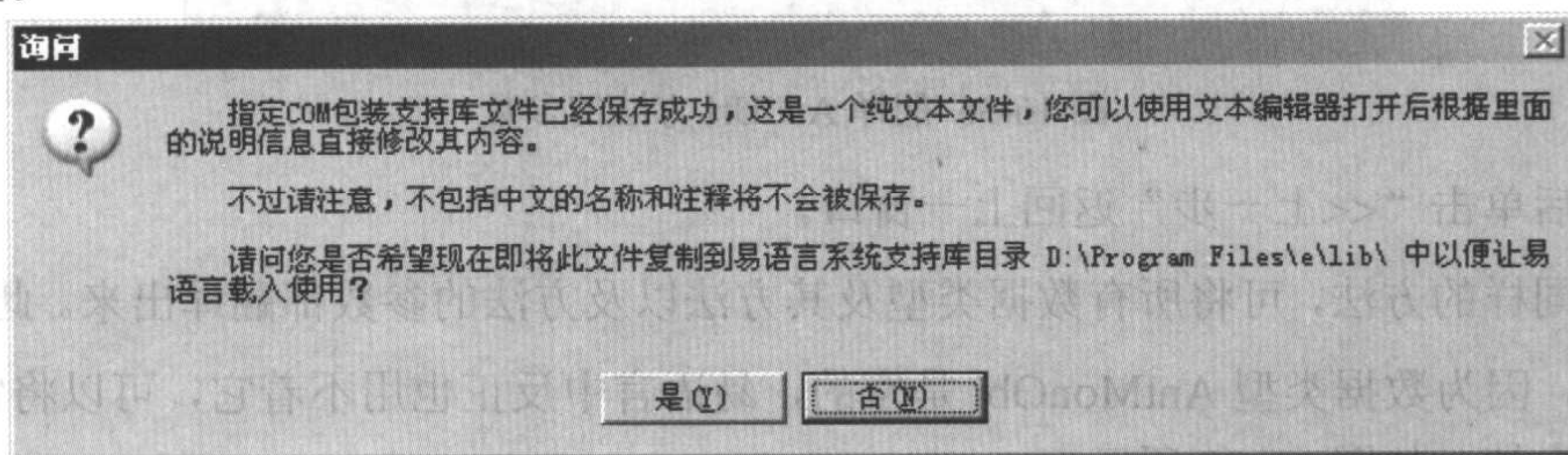
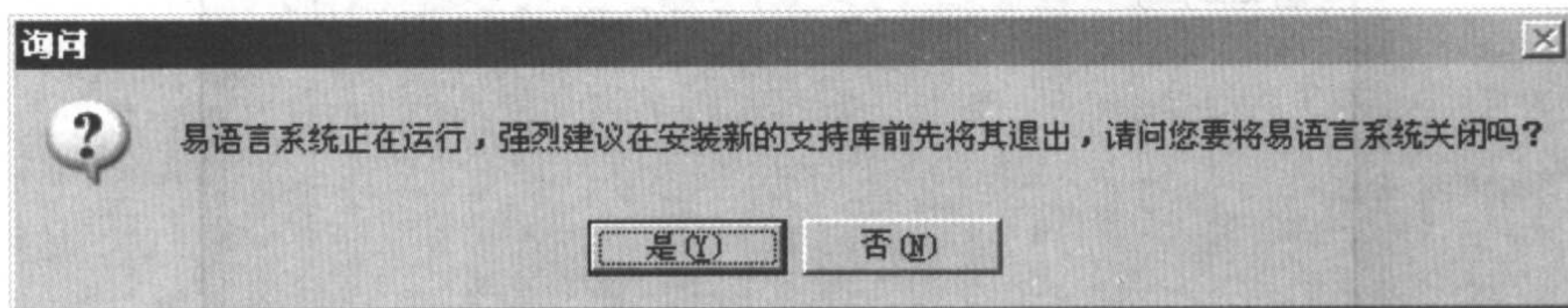


图 16-39 询问是否将包装文件复制到易系统目录

选择“是”。

如果易语言系统正在运行，还会有如下提示：



必须选择“是”，关闭运行中的易语言才能安装包装支持库成功。

最后弹出成功信息：



下面大家重新启动易语言，会在支持库列表中找到刚刚包装好的“网络蚂蚁类型库”，如图 16-40 所示。



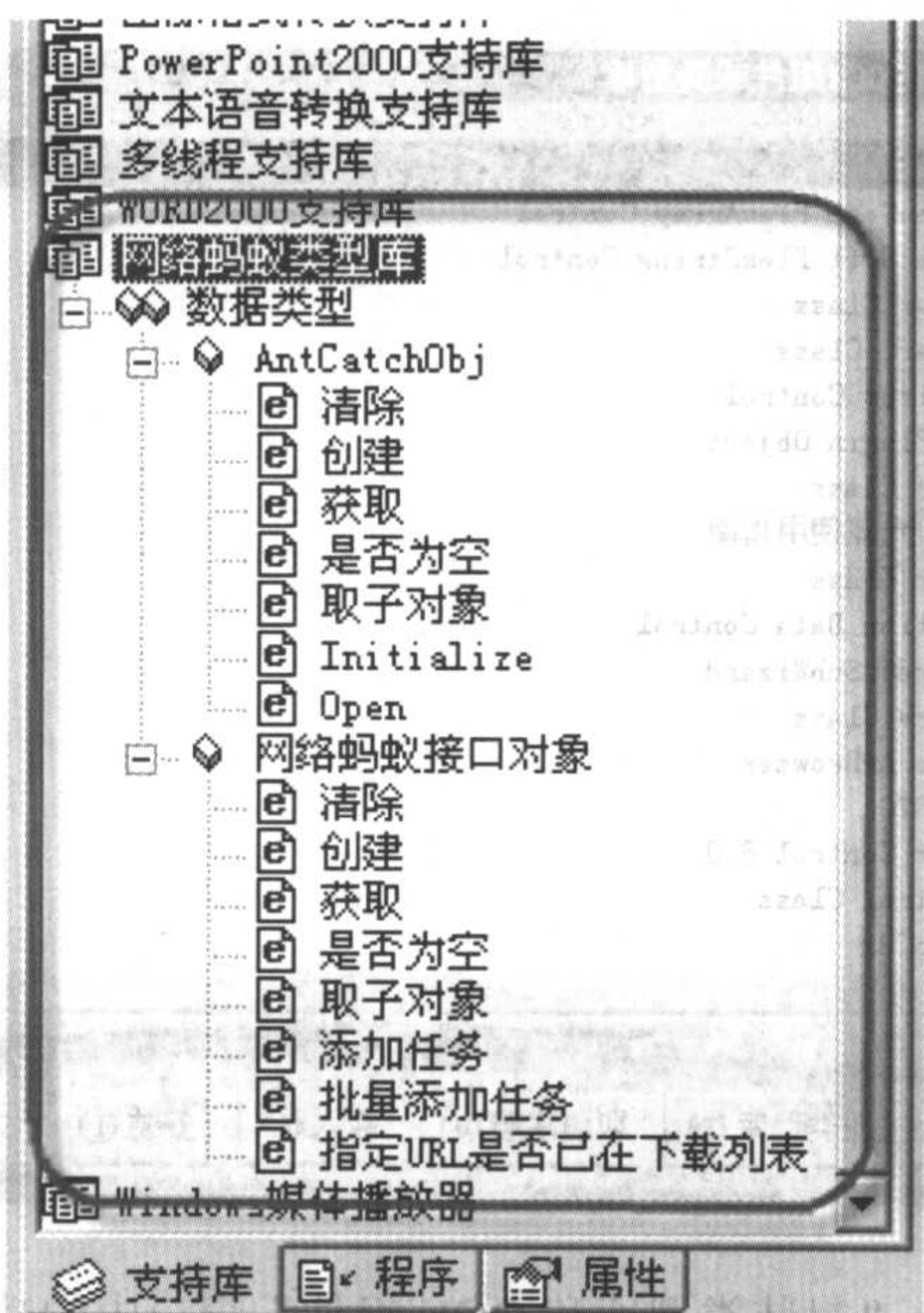


图 16-40 刚刚封装的“网络蚂蚁类型库”

大家至少有以下三点发现：

1. 易语言为每个数据类型自动增加了 5 个方法[注]：“清除 ( )”、“创建 ( )”、“获取 ( )”、“是否为空 ( )”、“取子对象 ( )”；
2. 数据类型“网络蚂蚁接口对象”因为在前面的汉化工作已经变成“全中文”，而 AntCatchObj 因为没有被汉化仍保留原有的英文信息；
3. AntMonObj 没有出现在数据类型列表中，是因为已把它的“输出”标志去掉了。

[注]：

当安装类型库时，易语言会为每个数据类型自动增加以下方法中的 0 个或多个：“清除 ( )”、“创建 ( )”、“获取 ( )”、“是否为空 ( )”、“取子对象 ( )”、“挂接事件 ( )”等。至于具体增加其中的哪几个方法，则视不同的数据类型而定。

在此重点说明其中的“挂接事件”方法：如果类型库中包含“事件组件”（安装后该事件组件图标将出现在“组件面板”中，与其他普通组件一样可摆放到设计窗口，它有一系列事件，供用户编写事件处理代码），易语言会自动为拥有该“事件组件”的数据类型添加“挂接事件”方法，该方法有一个类型为“相应事件组件”的参数。调用数据类型的“挂接事件”方法可将“事件组件”（包括其中的事件处理代码）“挂接”到该数据类型上。用户只有主动调用“挂接事件”方法，自己编写的事件处理代码才会正常工作。通常可在窗口的“创建完毕”事件中调用“挂接事件”方法。

至此，类型库封装的操作全部完成。后面将使用这个类型库编写一个简单的示例程序。

一些补充：

大家可能早就注意到，在图 16-31 的底部有一排按钮，而前面只使用了最右面的按钮“>>下一步”。图 16-41 中突出标识了这一排按钮：



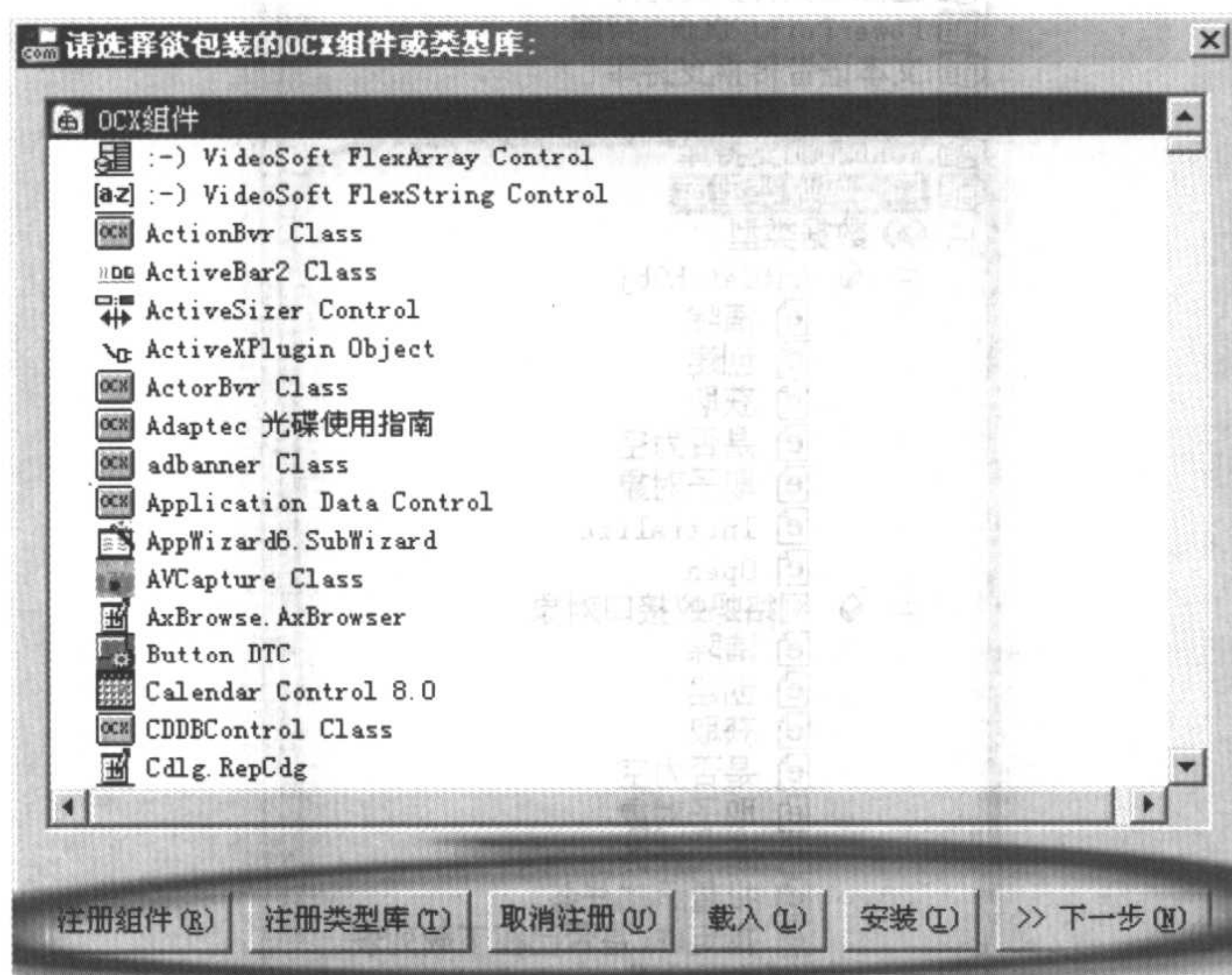


图 16-41 “封装类型库和 OCX 组件”窗口中的功能按钮

这几个按钮的主要功能如下：

按钮名称	功能描述
注册组件	将未注册的 OCX 组件向操作系统注册 (只有已注册的 OCX 组件才能供易语言封装后使用)
注册类型库	将未注册的类型库向操作系统注册 (只有已注册的类型库才能供易语言封装后使用)
取消注册	将已注册的 OCX 组件或类型库取消注册 (这一步可能导致某些程序不能运行, 需慎重)
载入	打开已有的*.npk 文件, 以编辑其中的汉化信息
安装	打开已有的*.npk 文件, 并将其安装到易语言中
>>下一步	封装类型库或 OCX 组件, 最终将生成*.npk 文件 这个文件也可拿给别人使用(安装或载入)

现在假设已经有一个人(根据前面详细介绍的操作步骤)制作出了一个\*.npk 文件(即 OCX/COM 包装支持库文件)并把它放到了网络上。大家怎么使用它呢? 很简单, 首先下载该文件, 然后单击图 16-41 中的“安装”按钮, 选择该文件, “确定”即可。当然有一个前提: 与该\*.npk 文件所对应的 OCX 组件或类型库已在操作系统中注册。

下载使用别人提供的\*.npk 文件的目的不言自明: 汉化的工作只要一个人辛苦一次就够了!

### 16.2.2 类型库的使用

下面使用这个“网络蚂蚁类型库”编写一个简单的示例程序, 完成网络蚂蚁中“添加任务”的功能。



首先新建一个易程序，设计窗体界面如图 16-42 所示。

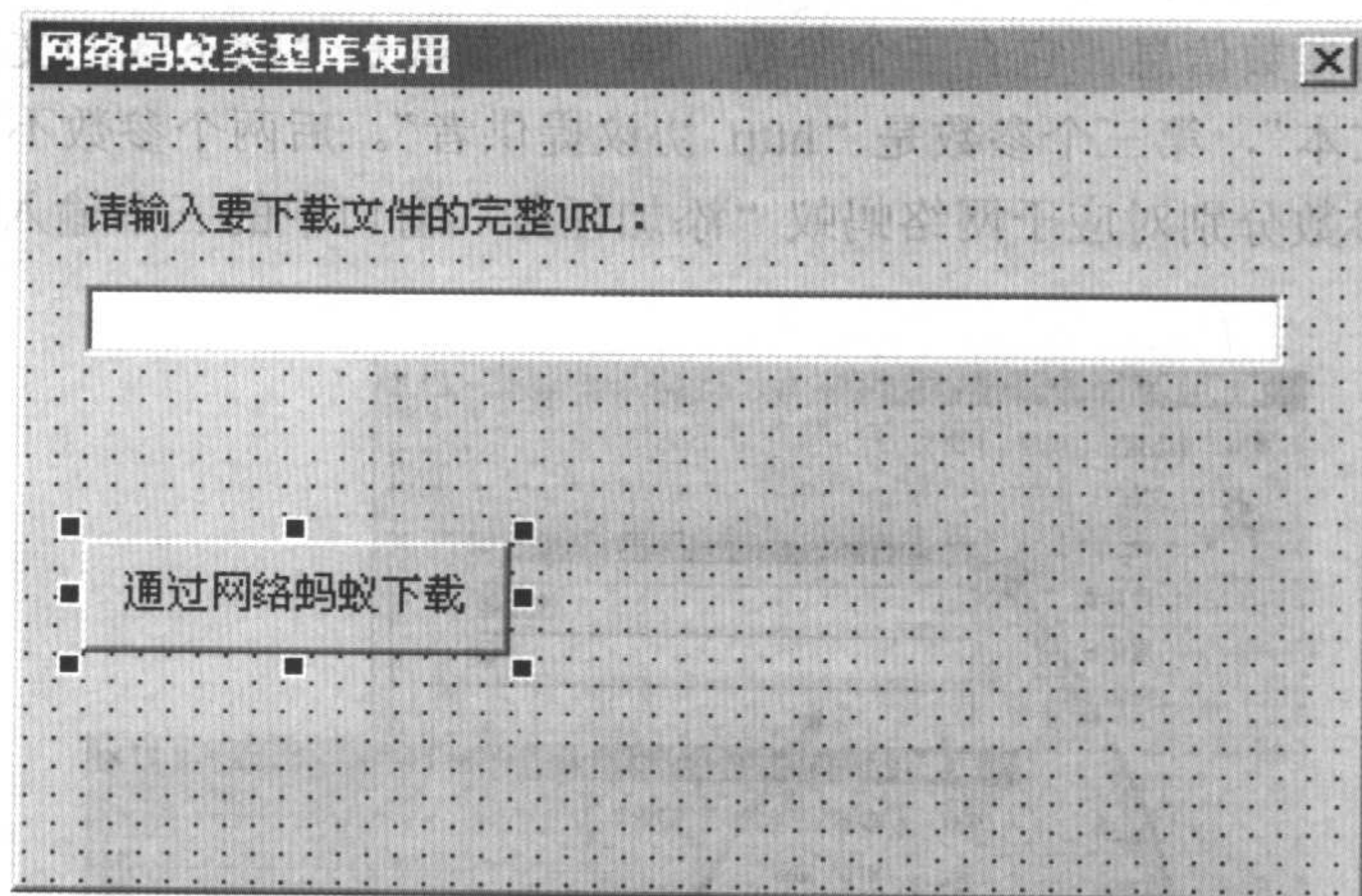


图 16-42 “网络蚂蚁类型库”设计界面

属性设置如下：

组件（或窗口）名称	属性名称	属性值
_启动窗口	标题	“网络蚂蚁类型库”
标签 1	标题	“请输入要下载文件的完整 URL”
编辑框 1		
按钮 1	标题	“通过网络蚂蚁下载”

接下来，双击“按钮 1”，进入代码编辑区，输入以下代码：

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
网络蚂蚁	网络蚂蚁接口对象			

```

如果真 (网络蚂蚁.指定URL是否已在下载列表 (编辑框1.内容) = 真)
    信息框 (“您指定的URL已在下载列表中，请勿重复下载！”，0，)
    返回 0
网络蚂蚁.添加任务 (编辑框1.内容, “”, “”)
    
```

简单解释一下上面的代码：

程序中定义了一个类型为“网络蚂蚁接口对象”的局部变量“网络蚂蚁”，所有功能都是调用这个变量的方法来完成的。类型“网络蚂蚁接口对象”是“网络蚂蚁类型库”中定义的数据类型。

程序的流程比较简单：首先调用方法“指定 URL 是否已在下载列表 ()”判断 URL（存于“编辑框 1.内容”中）是否已在下载列表中，如果已在下载列表中，则给出提示并返回；如果 URL 未在下载列表中，则调用方法“添加任务 ()”将该 URL 提交给网络蚂蚁进行下载。





这里有必要对“添加任务( )”方法进行说明。其功能是调用网络蚂蚁的“添加任务”窗口并传递相应的参数信息。它有三个参数，第一个参数是“欲下载的文件 URL”，第二个参数是“注释文本”，第三个参数是“http 协议提供者”。后两个参数不重要，可以提供空文本。这三个参数分别对应于网络蚂蚁“添加任务”窗口中的三个输入框，如图 16-43 所示。



图 16-43 “添加任务( )”方法的三个参数

程序已经编写完毕，请选择菜单“运行”→“运行”（或直接按 F5 键）运行该程序，在编辑框中输入下载地址“http://www.dywt.com.cn/edown/e/echins.exe”，单击窗口中的“通过网络蚂蚁下载”按钮，会发现网络蚂蚁将自动启动（如果之前尚未运行），并打开“添加任务”窗口，此时单击“确定”按钮将开始下载文件。如图 16-44、图 16-45 所示。

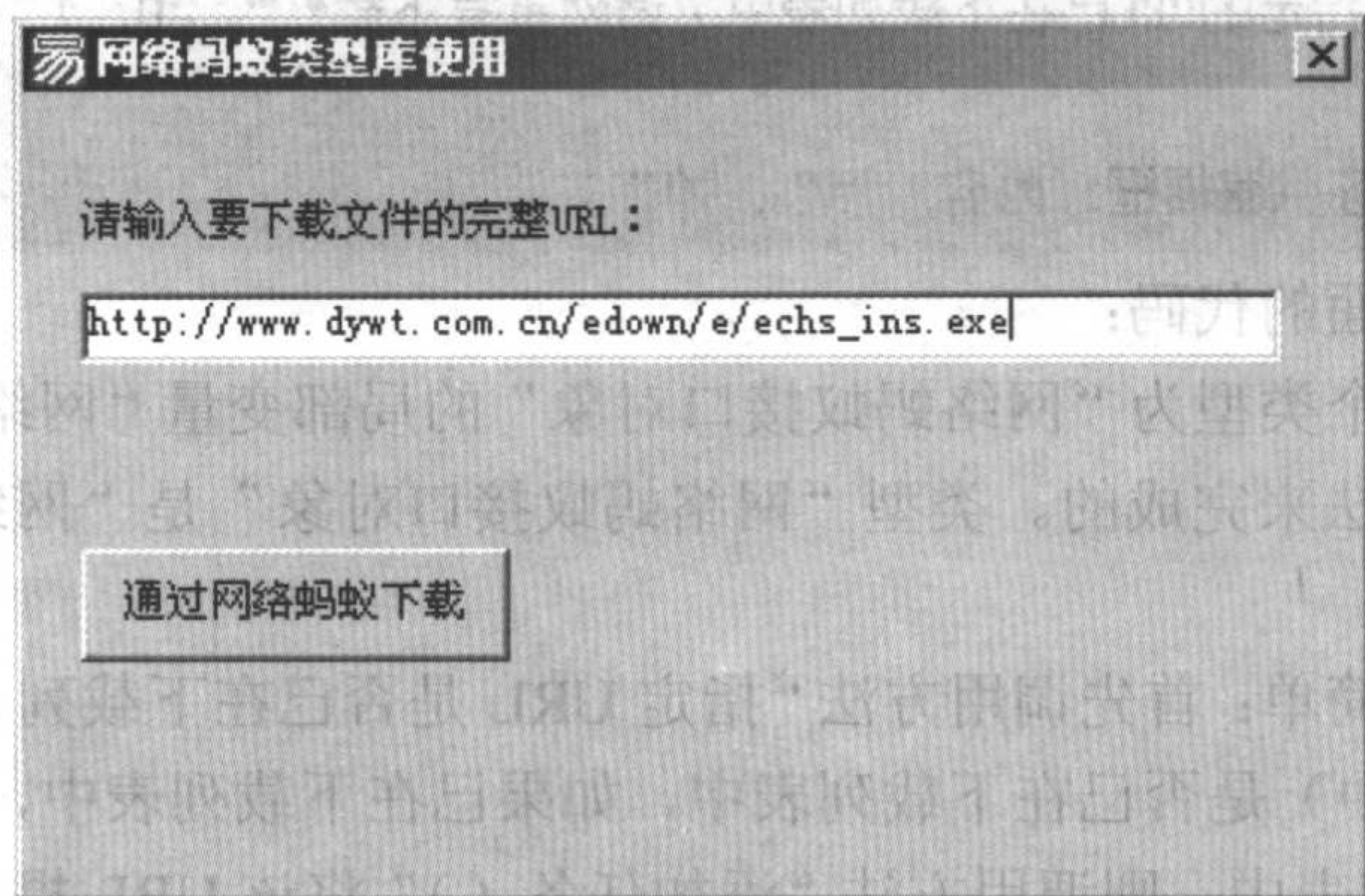


图 16-44 运行易程序



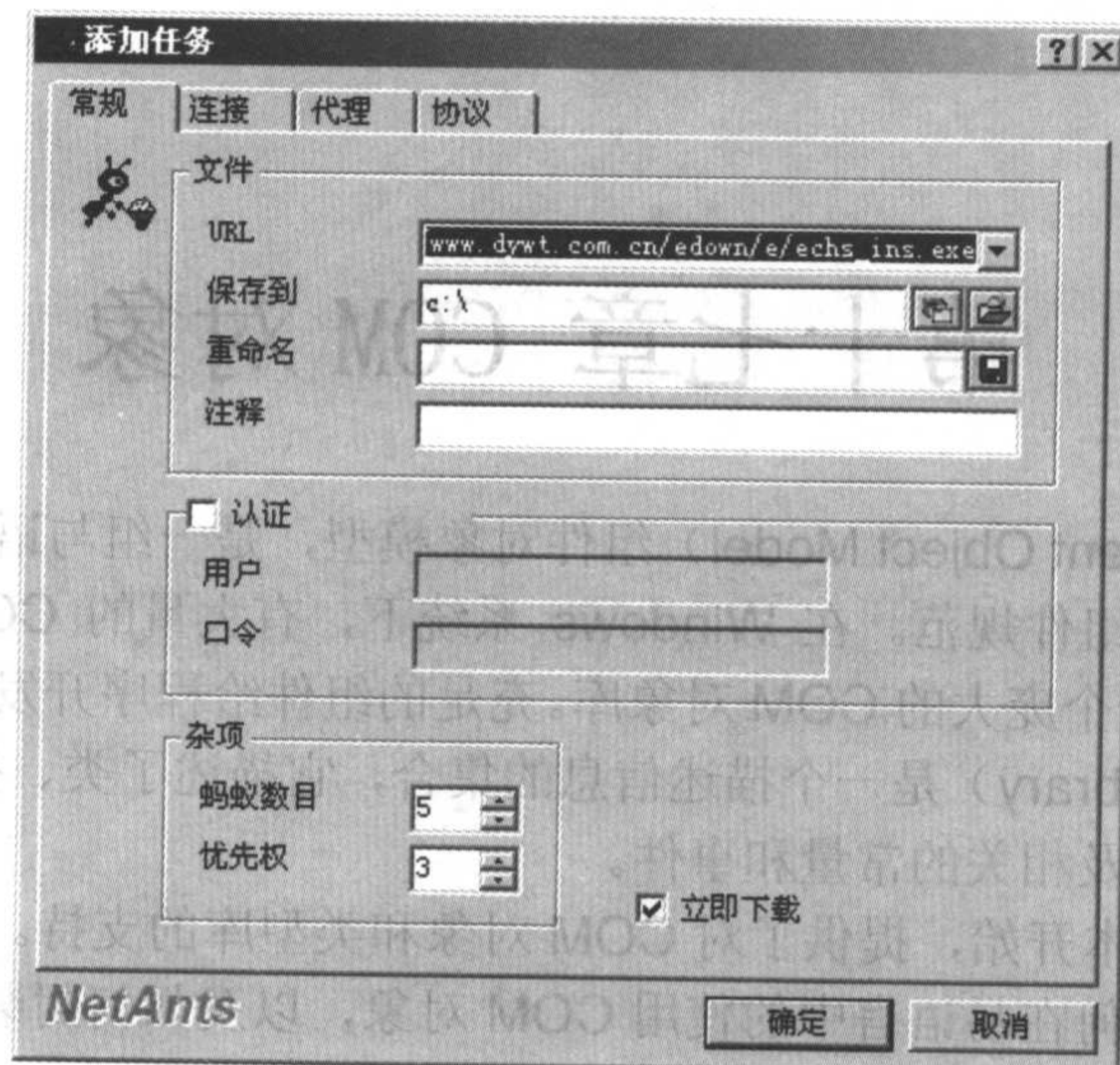


图 16-45 “网络蚂蚁类型库使用”运行效果

## 16.3 本章小结

易语言可以调用大量现成的 OCX 和类型库实现强大的功能，加快程序的编写速度。并且在使用中还可以汉化，使得编程更加容易。

大家还可以进行以下练习：

1. 按本章 OCX 的步骤制作 RM 播放器。
2. 按本章类型库的编程步骤调用网络蚂蚁下载文件。





## 第十七章 COM 对象

COM (Component Object Model) 组件对象模型, 是一组与语言无关的、二进制的、面向对象的、可复用组件规范。在 Windows 系统下, 有大量的 COM 对象存在, 尤其是 Windows 本身就是一个庞大的 COM 对象库。充足的组件给程序开发人员提供了极大便利。

类型库 (Type Library) 是一个描述信息的集合, 它描述了类、接口、接口方法、接口方法的参数类型, 以及相关的常量和事件。

易语言从 3.7 版本开始, 提供了对 COM 对象和类型库的支持。

本章主要介绍如何在易语言中的使用 COM 对象, 以及如何封装和使用类型库。

### 17.1 基本概念

#### 17.1.1 什么是 COM

COM (Component Object Model), 组件对象模型, 是一组与语言无关的、二进制的、面向对象的、可复用组件规范。

#### 17.1.2 COM 对象

在 COM 规范中, 没有 COM 对象的严格定义, COM 组件提供给客户的是以对象形式封装起来的实体, 客户与组件交互的实体是 COM 对象。COM 对象有自己的属性和方法, 但这些都 COM 封装了起来, 客户只有通过接口才能对 COM 的方法进行调用, 接口是 COM 与外界通信, 交互的唯一途径。

COM 对象的概念有点类似于 C++ 中对象的概念。

#### 17.1.3 COM 接口

COM 接口是 COM 规范中最重要的部分, COM 规范的核心内容就是对接口的定义, 甚至可以说“在 COM 中接口就是一切”。组件与组件之间、组件与客户之间都要通过接口进行交互。接口成员函数将负责为客户或其他组件提供服务。与标识 COM 对象的 CLSID 类似, 每一个 COM 接口也使用一个 GUID 来进行标识, 该标识也被称为 IID (Interface Identifier, 接口标识符)。



## 17.2 COM 对象的使用

### 17.2.1 新的数据类型“对象”

易语言从 3.7 版开始，增加了一个新的数据类型“对象”（位于系统核心支持库），通过它就可以方便地引用 COM 对象。图 17-1 就是在易语言工作夹“支持库”面板中看到的“对象”及其所有方法。

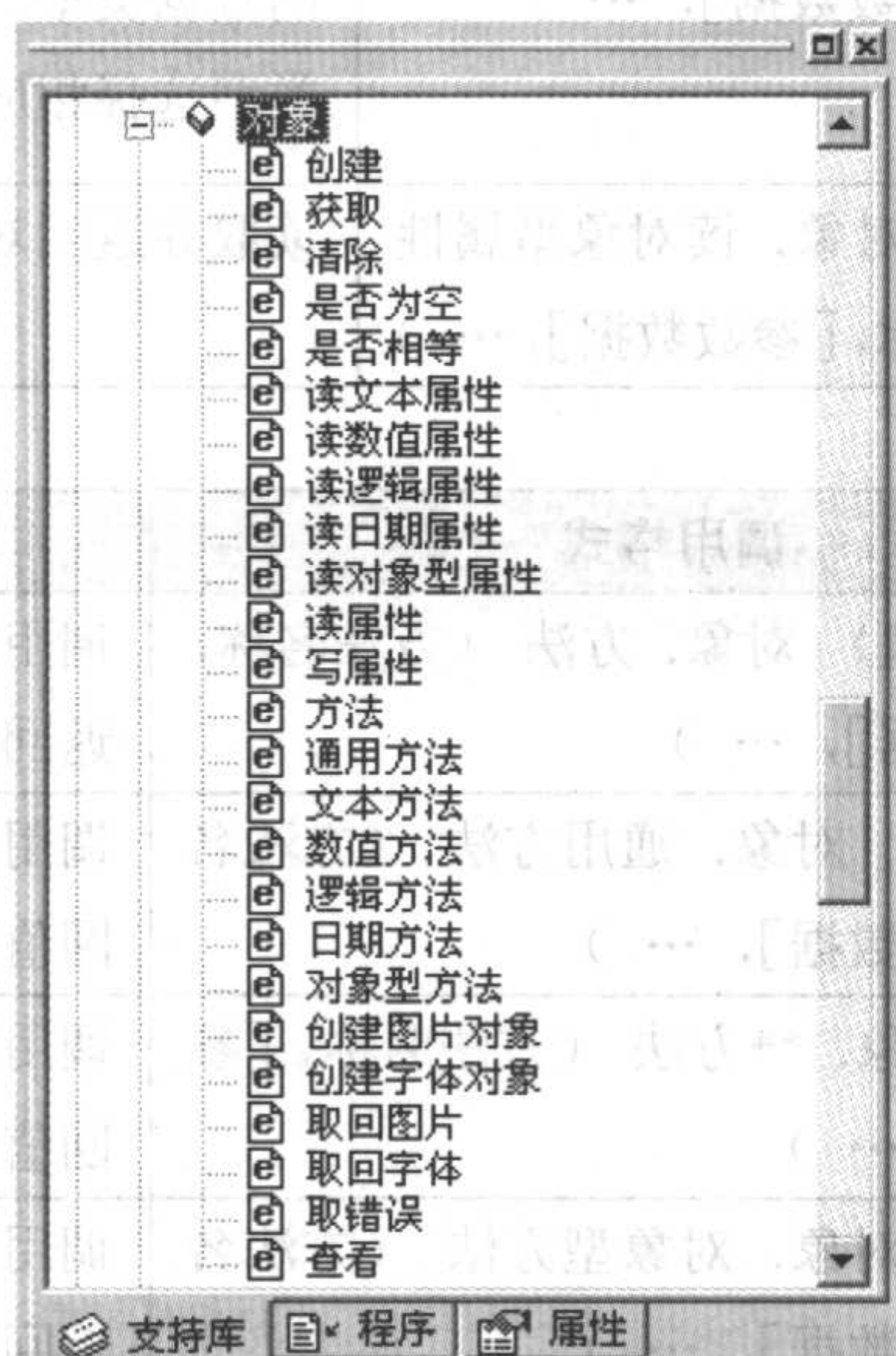


图 17-1 “对象”的方法

“对象”类型中，比较常用的方法有：

#### 1. 创建清除类

方法名称	调用格式	解释
创建	〈逻辑型〉 对象. 创建 (对象类型, [类型库文件名])	创建指定类型的 COM 对象，本对象中的原有内容将被释放。
获取	〈逻辑型〉 对象. 获取 (对象类型)	获取当前操作系统中已经存在的指定类型 COM 对象，本对象中的原有内容将被释放。
清除	〈无返回值〉 对象. 清除 ( )	将本对象的内容释放并清空。如果不调用本方法，则对象在退出其作用区域时会自动被释放。例如：假设对象存在于子程序局部变量中，当子程序调用退出时，该对象会被自动释放。





## 2. 属性读写类

方法名称	调用格式	解释
写属性	〈逻辑型〉 对象. 写属性 (属性名称, [参数数据], ...)	为对象的指定属性赋值, 提供的属性值可为各种数据类型。
读属性	〈变体型〉 对象. 读属性 (属性名称, [参数数据], ...)	读取并返回对象的指定属性值, 可以用作读取任意类型的属性。注: 读对象属性时, 第二个及以后的参数可以为空, 下同
读**属性	〈**〉 对象. 读属性 (属性名称, [参数数据], ...)	读取并返回对象的指定属性值, 返回相应的数据类型。应用中, 应根据属性的实际类型选择相应的读属性方法。
读对象型属性	〈对象〉 对象. 读对象型属性 (属性名称, [参数数据], ...)	读取并返回对象的指定对象属性值。

## 3. 方法调用类

方法名称	调用格式	解释
方法	〈无返回值〉 对象. 方法 (方法名称, [参数数据], ...)	调用对象的指定方法, 该方法无返回值。
通用方法	〈变体型〉 对象. 通用方法 (方法名称, [参数数据], ...)	调用对象的指定方法, 该方法返回值类型为“变体型”。
**方法	〈**〉 对象. **方法 (方法名称, [参数数据], ...)	调用对象的指定方法, 该方法返回值类型为相应数据类型。
对象型方法	〈对象〉 对象. 对象型方法 (方法名称, [参数数据], ...)	调用对象的指定方法, 该方法的返回值类型为“对象”。

## 4. 其它类

方法名称	调用格式	解释
查看	〈无返回值〉 对象. 查看 ()	通过对话框的方式查看本对象的调用格式信息, 便于编写相关程序。仅在易程序的调试版本中被执行, 在发布版本中将被直接跳过。该方法直到对话框关闭才返回。
取错误	〈文本型〉 对象. 取错误 ()	当读写对象属性、执行对象方法或取回字体时, 紧跟该语句后执行本方法可以检查其是否执行成功。如果成功, 本命令将返回空文本, 如果失败, 本命令将返回一个描述具体错误信息的非空文本。
是否为空	〈逻辑型〉 对象. 是否为空 ()	如果本对象的内容为空, 返回真, 否则返回假。
是否相等	〈逻辑型〉 对象是否相等 (欲检查的对象)	如果本对象的内容与指定对象的内容相等, 返回真, 否则返回假。



在“对象”的所有方法中，“查看（）”是一个很特殊的方法。它太奇怪了，完全不同于其他任何普通方法。易语言的开发者们通过提供这样一个与众不同的对象方法，显示出了他们无穷的创意——当然也得益于 COM 规范的独特设计。

说“查看（）”方法特殊，是因为它即不对最终程序产生任何影响，又不具备调试功能。它实际上是一个“针对程序开发者的”动态帮助系统，仅在“易程序的调试版本”中有效，在发布版本中将被直接跳过。当调试运行易程序并执行到“对象.查看（）”方法时，将弹出如图 17-2 所示的对话框。

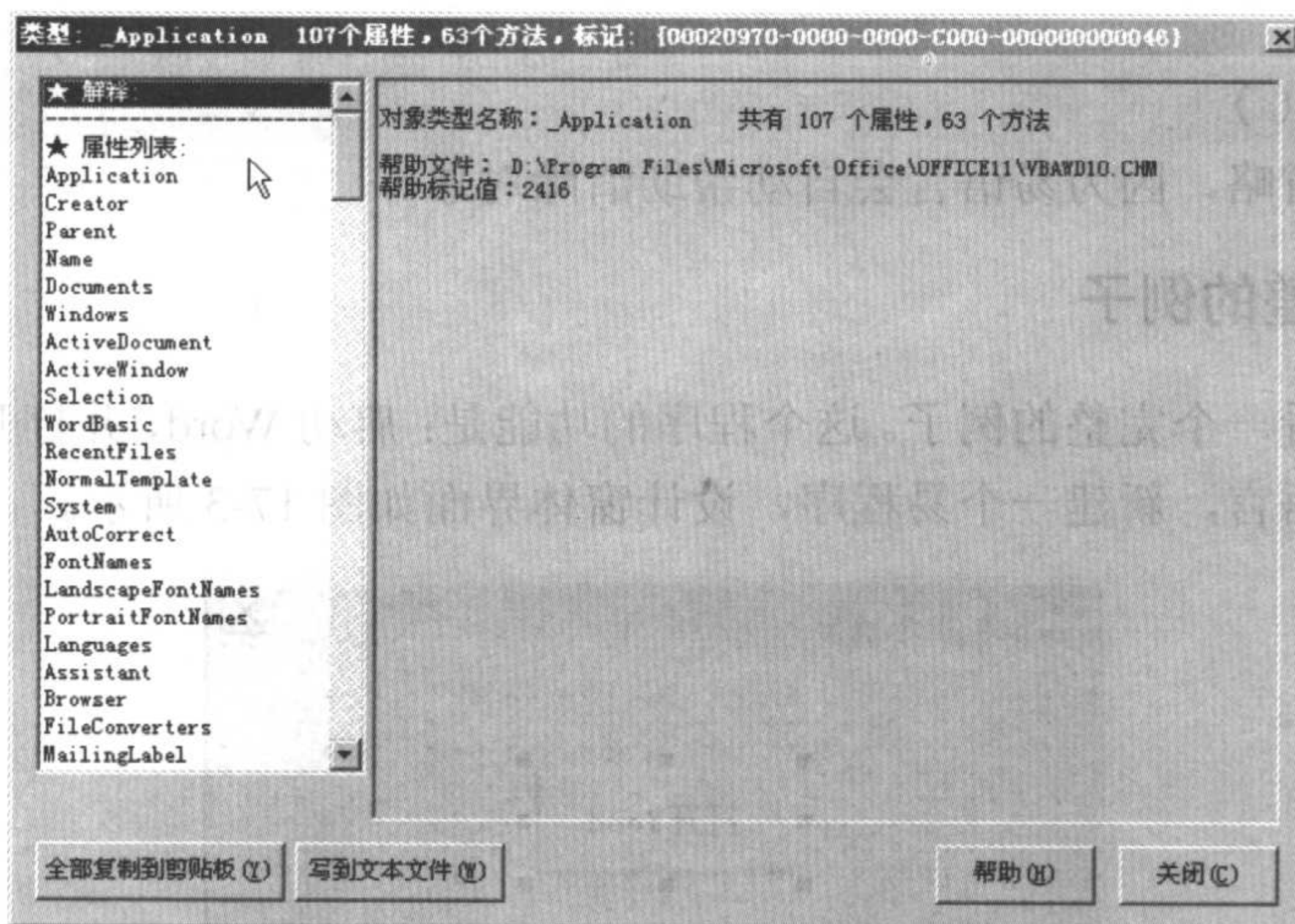


图 17-2 “对象.查看（）”执行效果图

可以看到在此对话框中，列出了该对象的所有基本信息，包括属性列表、方法列表，以及各属性、方法在易语言中的调用方式。有了这些信息，即使没有其他参考资料，也能大致了解该对象的使用方法。

### 17.2.2 使用 COM 对象的一般步骤

第一步：定义“对象”类型的变量

如下面定义了一个名称为“Word 对象”的变量，其类型为“对象”：

变量名	类型	静态	数组	备注
Word对象	对象			

第二步：创建或获取 COM 对象

如下面的代码创建了一个 Word 对象：

```
Word对象.创建 (“Word.Application”, )
```

在这里“Word.Application”是“对象类型”名称，它由微软（Microsoft）定义，类似的还有“Excel.Application”“PowerPoint.Application”“Access.Application”等。

第三步：查看对象的可用属性和方法

即调用：





对象名.查看()

如果对要操作的对象已非常熟悉,或有足够的参考资料,可以省略这一步。

第四步:读写对象属性,调用对象方法

下面是两个例子:

Word对象.写属性 (“Visible”, 真)

文档对象.方法 (“Open”, 取运行目录 () + “\e.doc”)

第五步:清除对象

即调用:

对象名.清除()

这一步可以省略,因为易语言会自动帮助清除对象。

### 17.2.3 一个完整的例子

下面大家来看一个完整的例子。这个程序的功能是:启动 Word,并打开一个 Doc 文档。首先打开易语言,新建一个易程序,设计窗体界面如图 17-3 所示。

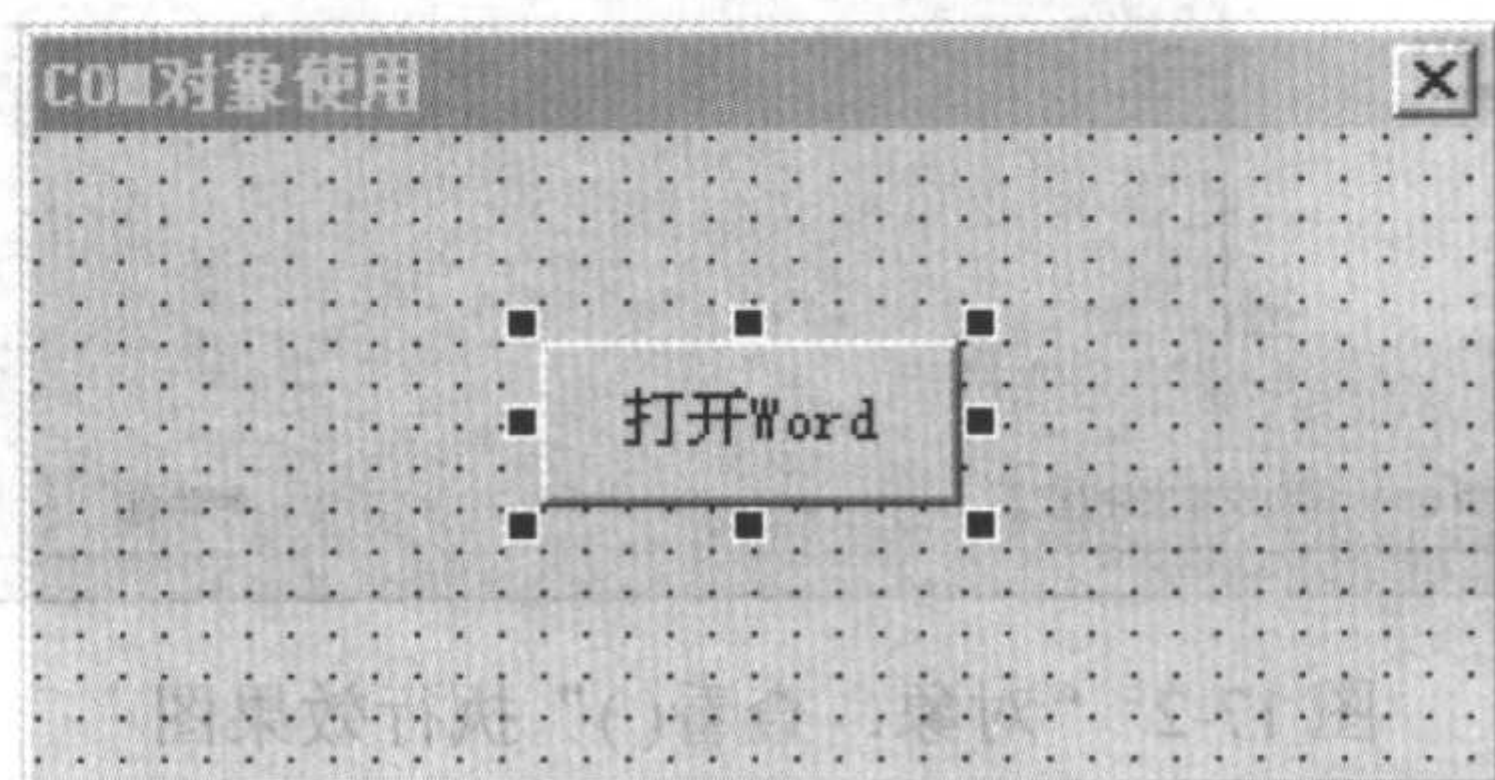


图 17-3 “COM 对象使用”设计界面

属性设置如下:

组件(或窗口)名称	属性名称	属性值
_启动窗口	标题	“COM 对象使用”
按钮 1	标题	“打开 Word”

接下来,双击“按钮 1”,进入代码设计工作区,输入以下代码:

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
Word对象	对象			
文档对象	对象			

Word对象.创建 (“Word.Application”, )

’ Word对象.查看 ()

Word对象.写属性 (“Visible”, 真)

文档对象 = Word对象.读对象型属性 (“Documents”, )

’ 文档对象.查看 ()

文档对象.方法 (“Open”, 取运行目录 () + “\e.doc”)

在上面的代码中,首先定义了两个“对象”型变量:“Word 对象”和“文档对象”。



然后调用“对象.创建( )”方法生成“Word 对象”:

Word 对象.创建(“Word. Application”,)

接着设置“Word 对象”的 Visible (可视) 属性为真 (因为其刚创建后默认为假), 使 Word 显示出来:

Word 对象.写属性(“Visible”,真)

然后读取“Word 对象”的“Documents”属性, 得到“文档对象”:

文档对象 = Word 对象.读对象型属性(“Documents”,)

最后, 调用“文档对象”的“Open”方法, 打开指定的 Doc 文档:

文档对象.方法(“Open”, 取运行目录() + “\e.doc”)

另外在程序编写过程中, 还用到了“对象.查看( )”方法, 因为已经用不着了, 所以使用快捷键: Ctrl+K, 将其屏蔽:

‘Word 对象.查看 ()

‘文档对象.查看 ()

程序已经编写完毕, 请选择菜单“运行”→“运行”(或直接按 F5 键)运行该程序, 单击窗口中的“打开 Word”按钮, 会发现 Word 已自动启动, 并打开了指定的 doc 文档, 如图 17-4 所示。

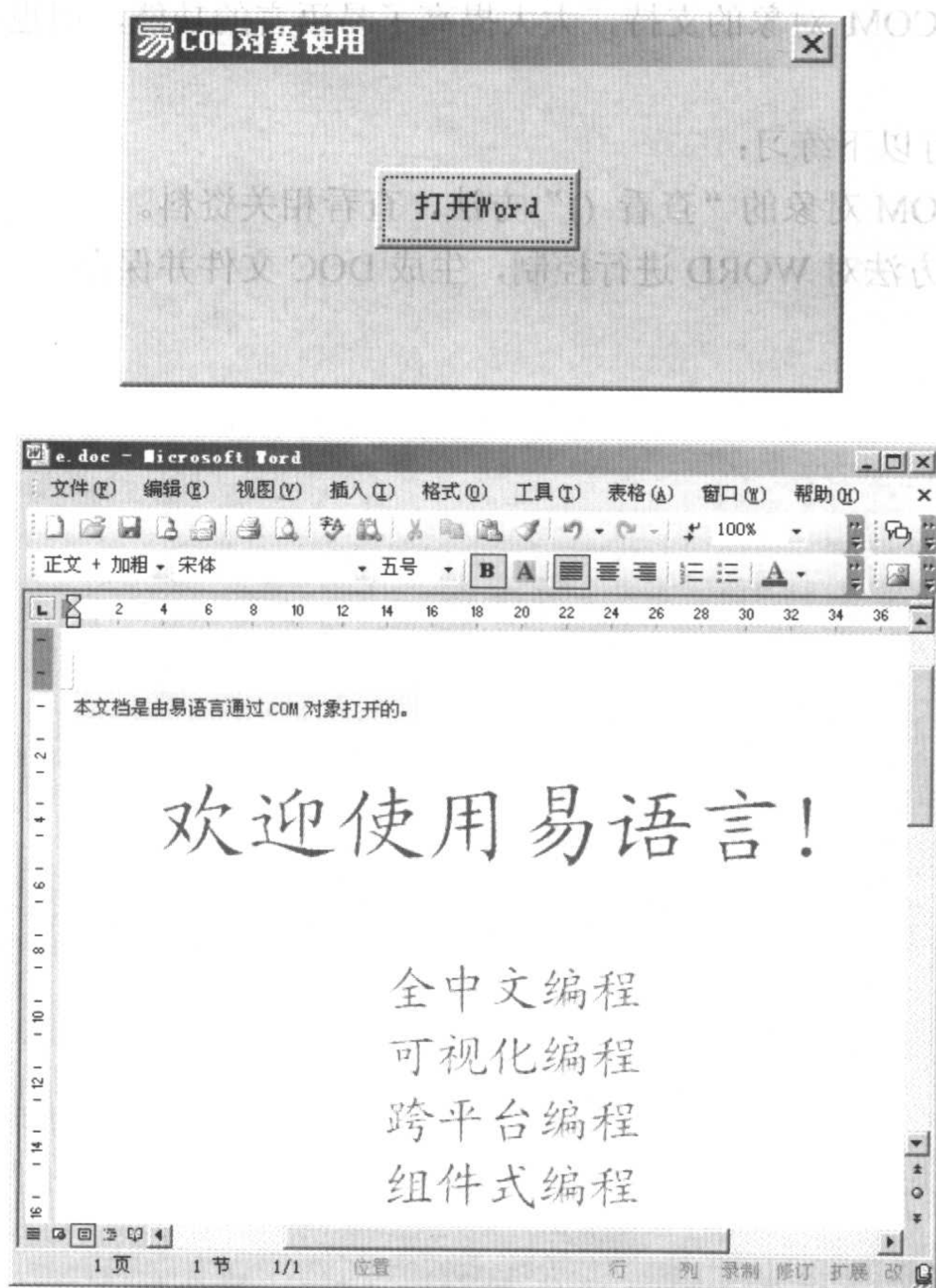


图 17-4 “COM 对象使用”执行效果图





## 17.3 本章小结

在本章中，详细介绍了在易语言中通过新的数据类型“对象”使用 COM 对象的方法。大家都看到了，仅仅使用“对象”，就可以把 COM 对象“玩弄于股掌之间”，实在是了不起的事情。

大家还可以看到，使用这种方式调用 COM 对象，虽然通用性强，但无论使用 COM 中定义的何种数据类型（如前面例子中的 Documents），都必须绕道经过“对象”来完成；而且在“对象”中访问具体类型的“属性”和“方法”的方式，也大异于以前。

您会发现，在易语言中使用 COM 对象的方法是很容易的。然而具体到某个 COM 对象的应用，却又不是那么简单了。像 Microsoft 提供的 Office 系列组件，其对象层次相当复杂，虽然也有大量的中文资料可以查询，但要在短期内研究透彻也不是很容易的事情。况且有的 COM 对象只提供英文资料，甚至没有资料，使用起来就更困难了。

易语言通过对 COM 对象的支持，大大提高了易语言的功能，但也在一定程度上加大了应用难度。

大家也可以进行以下练习：

1. 学习使用 COM 对象的“查看 ()”方法，查看相关资料。
2. 使用 COM 方法对 WORD 进行控制，生成 DOC 文件并保存。



## 第十八章 易语言面向对象编程

面向对象编程是当前最流行的编程方式，支持面向对象编程语言也很多，常见的有 VB、C++、JAVA 等等，这些面向对象的编程语言都是英文编程语言。易语言在全中文编程的基础上，从 3.8 版推出后，也开始全面支持面向对象编程。

本章将先结合几个小例子，介绍面向对象编程的一些基本概念，最后将学习使用易语言编写一个面向对象编程的程序。

### 18.1 基础知识

#### 18.1.1 类的概念

要了解面向对象编程首先要了解类的概念。而类和对象是两个截然不同的概念，所以不要将类和对象混淆。

类可以看做一个活跃的类型，这种类型由编程者自己定义，定义该类型的内部结构，其中包括定义该类型中数据和行为。该类型的行为称为该类的方法；该类型中用来存放各种数据的变量，是该类的私有成员。

上面的定义可能比较抽象而难于理解，类也可以简单的看成是封装了各种数据和方法模块，但是类具有很多特有的个性，也是面对对象编程的重要组成部分。

对象是类的一个实例。如果将对象比做房子，那么类就是房子的设计图纸。所以面向对象编程的重点是类的设计，而不是对象的设计。

在易语言中，要新建一个类，可以通过菜单“插入”→“类模块”来实现，如图 18-1 所示。

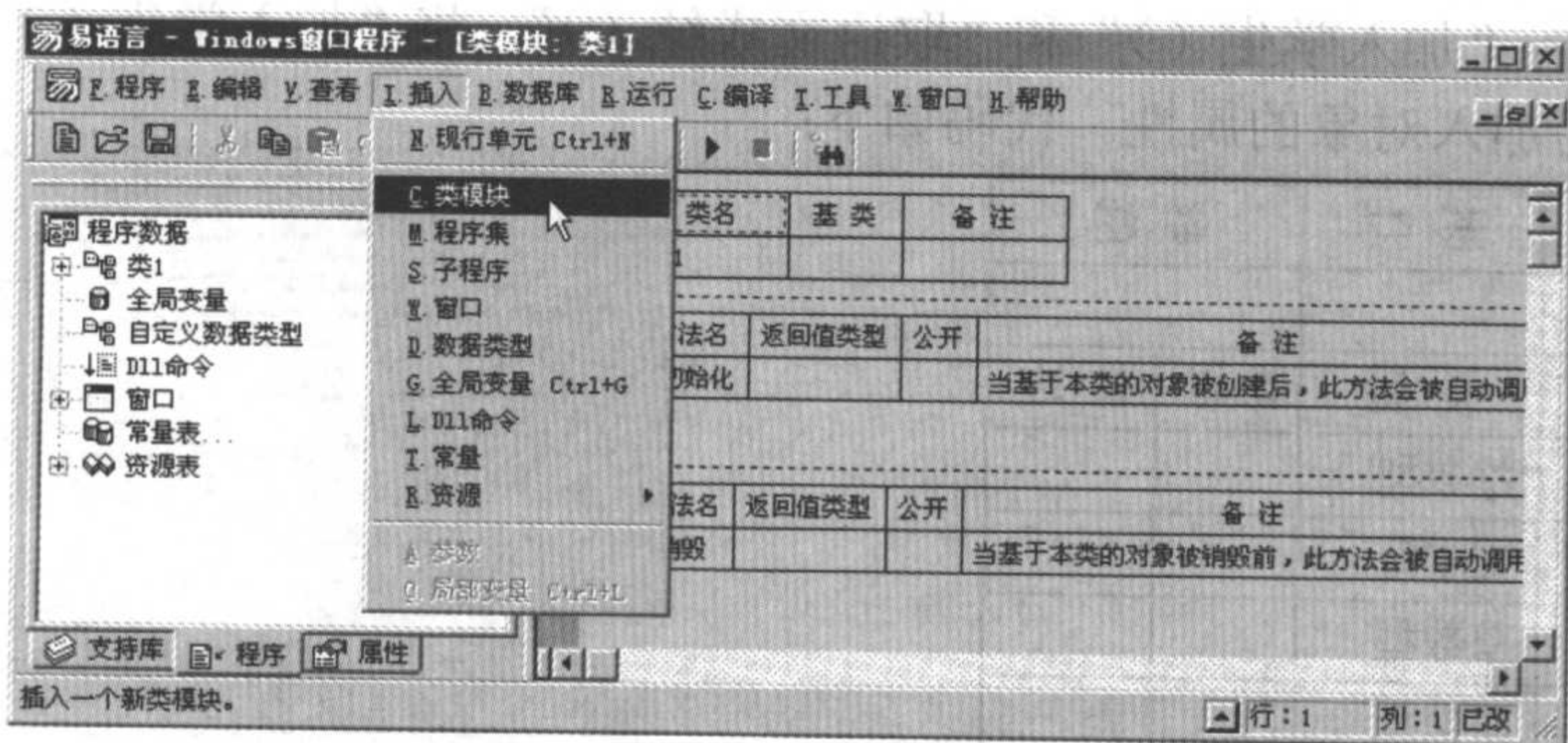


图 18-1 新建类





也可以通过在程序面板中的“程序数据”项目上点击鼠标右键，在弹出菜单中选择“新类模块”选项，如图 18-2 所示。

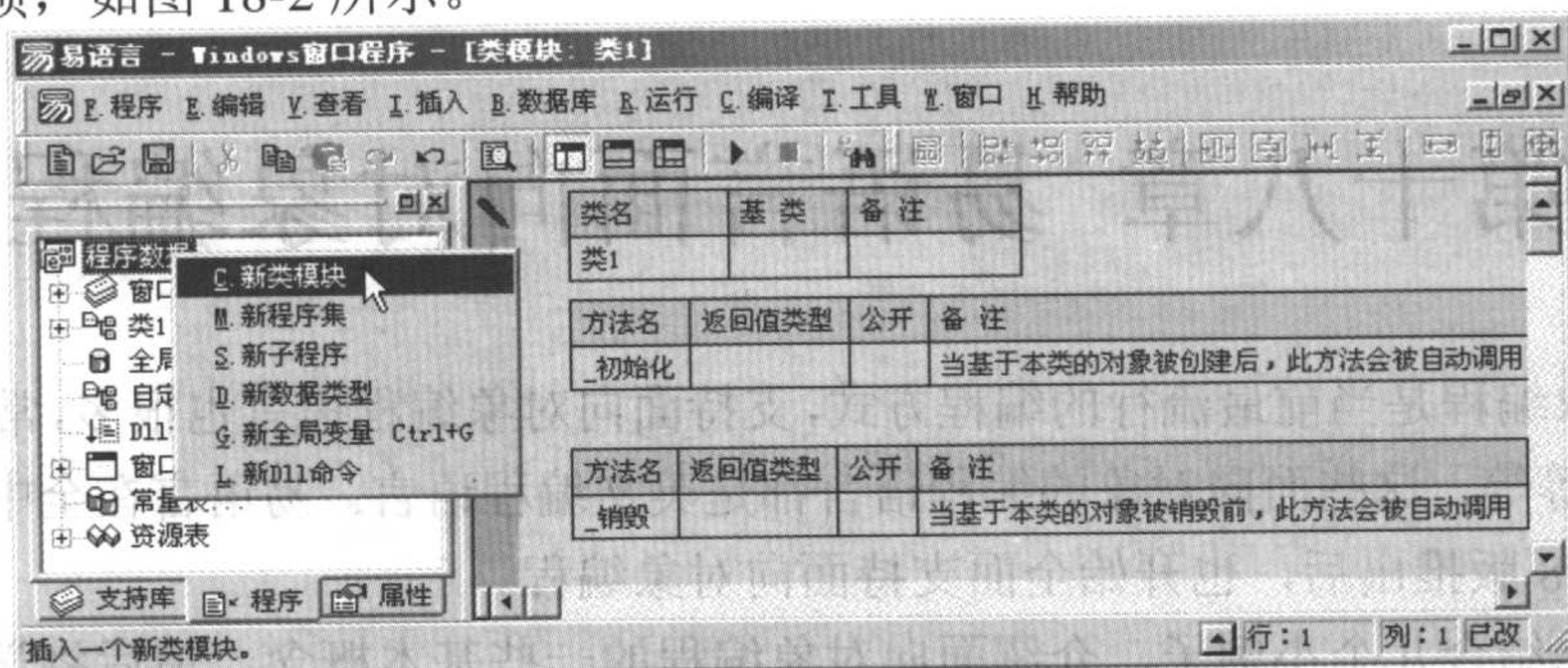


图 18-2 新建类模块

插入后的类，会自动生成“\_初始化 ()”方法和“\_销毁 ()”方法，这两个方法会在基于该类的对象创建和销毁时自动运行。

这两个方法，相当于其他面向对象的编程语言中的构造函数和析构函数，用来初始化对象和销毁对象。例如，可以在“\_初始化 ()”中，可以设置各私有成员的初始值；可以在“\_销毁 ()”方法中，编写一些需要收尾的工作，如关闭文件、关闭数据库等等。

其他很多面向对象编程语言中构造函数和析构函数都是由程序员手动编写的。易语言中自动生成这两个函数，无疑减轻了程序员们的负担。

## 18.1.2 类和对象的关系

可能上面对类和对象的描述还比较模糊，下面就举个实例来进一步了解类与对象的概念。“学生”是一类人，在校的学生有什么特点呢？学生有“学号”、“姓名”、“语文成绩”、“数学成绩”等等，单独拿出“学生”这个类，是个抽象的群体，所以想象不出“学生”这类人的具体模样身高等等实际的内容。“张三”是个学生，张三属于学生这个群体，他具有学生这类群体的所有特征，所以，我们说张三是“学生”类的一个实例，也就是说张三是基于“学生”类的一个对象。

下面就使用易语言创建一个名为“学生”的类。

易语言中，可以在新建类后按下 Ctrl+L 键，来给该类插入一个新方法；可以通过在类名上点击 Enter 键，给该类加入私有成员，类的私有成员相当于类的属性。

将新建的学生类中加入“学号”、“姓名”、“语文成绩”、“数学成绩”这几个属性，然后加入两个方法：“加入学生 ()”和“取语文成绩 ()”，将“加入学生 ()”方法新建 4 个参数，用来填写加入对象的属性。代码如下：

类名	基类	备注	
学生			
私有成员名	类型	数组	备注
学号	整数型		
姓名	文本型		
语文成绩	整数型		
数学成绩	整数型		



方法名	返回值类型	公开	备注		
加入学生		✓			
参数名	类型	参考	可空	数组	备注
加入学号	整数型				
加入姓名	文本型				
加入语文成绩	整数型				
加入数学成绩	整数型				

学号 = 加入学号

姓名 = 加入姓名

语文成绩 = 加入语文成绩

数学成绩 = 加入数学成绩

方法名	返回值类型	公开	备注
取语文成绩	整数型	✓	

返回 (语文成绩)

程序中，“加入学生 ()”方法用来加入新学生的一系列的属性，“取语文成绩 ()”方法用来取得学生的语文成绩。当然如果需要，可以加入更多的方法，如“取数学成绩 ()”、“置语文成绩 ()”等等。

为了达到信息隐藏的目的，即让类仅仅公开必须要让外界知道的内容，而隐藏其他一切内容。类的方法有“公开”属性，设置“公开”属性的方法后，当基于该类的对象被调用时，才可以显示和调用，而其他的方法将被隐藏。被隐藏的方法可以在类的内部和其派生类中调用，这个后面将会讲到。

那如何才能创建一个基于“学生”类的对象呢？例如加入一个学生“张三”，在程序中，可以新建一个对象型变量，将变量的类型定义为“学生”，使用“对象.加入学生 ()”，就可以加入一个新学生。代码如下：

窗口程序集名	保留	备注	
窗口程序集1			
变量名	类型	数组	备注
学生集合	学生	0	

子程序名	返回值类型	公开	备注
_加入学生按钮_被单击			

变量名	类型	静态	数组	备注
新学生	学生			

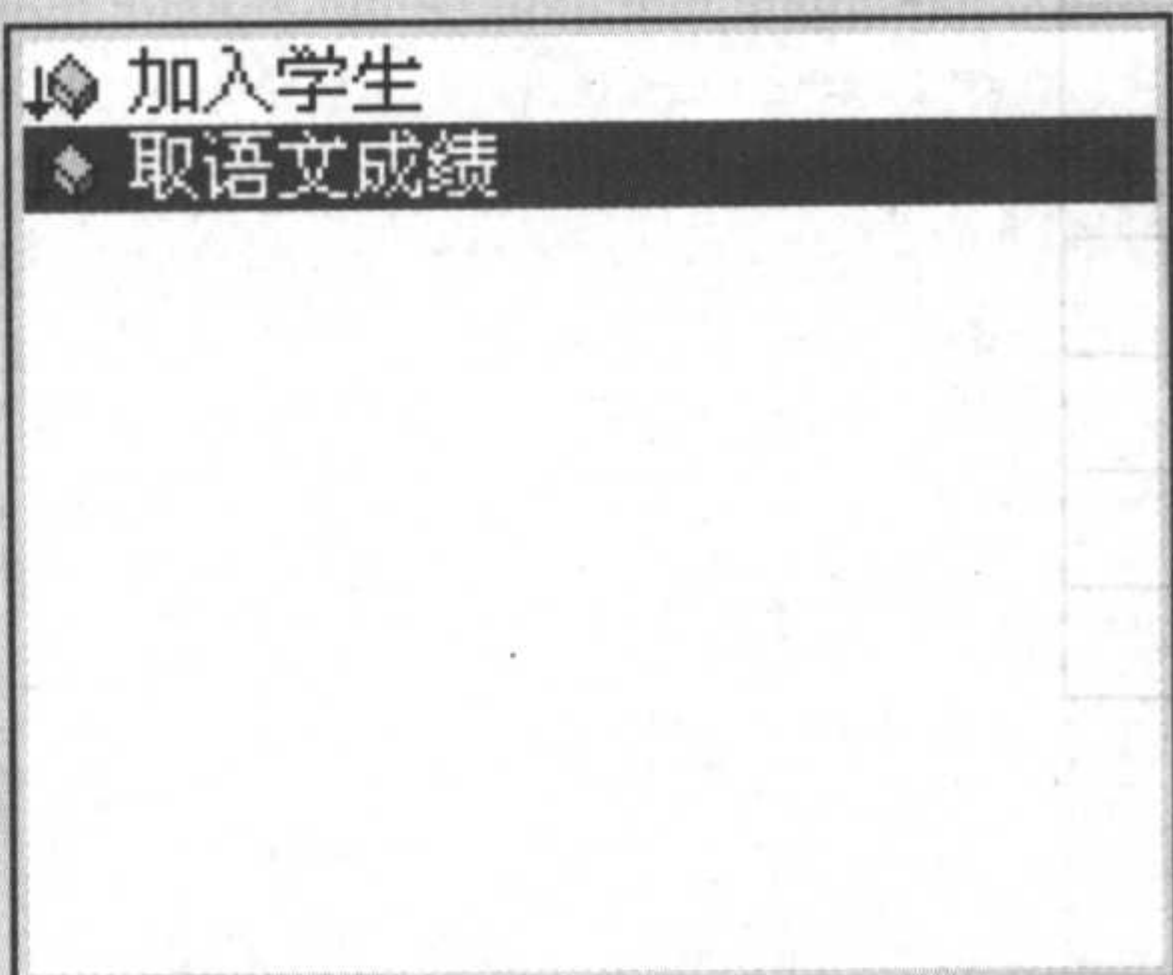
新学生.加入学生 (1, “张三”, 100, 99)

加入成员 (学生集合, 新学生)





新学生.



上述程序中，对象输入后智能语法提示只列出了公开的方法。上述程序实现加入一个新的学生“张三”，并设置了“张三”的所有属性，最后将张三加入到“学生集合”变量中。

### 18.1.3 类中私有成员的特性

类的私有成员，是不可以直接更改和调用的，例如直接用赋值语句“对象.属性=XX”，这样直接赋值是不允许的，这也是出于保护数据的目的。所以，要想读取或赋值私有成员，就必须通过新建专门的方法来实现，即对象调方法，方法改属性。所以上面的“学生”对象中加入了“取语文成绩（）”方法，专门用来取出对象的语文属性值。

例如，在上面的“学生”类中再新建一个“置语文成绩（）”的方法，代码如下：

方法名	返回值类型	公开	备注
取语文成绩	整数型	✓	

返回（语文成绩）

方法名	返回值类型	公开	备 注		
置语文成绩		✓			
参数名	类 型	参考	可空	数组	备 注
实际成绩	整数型				

语文成绩 = 实际成绩

这样，要想更改“张三”的语文成绩，就可以使用代码：

子程序名	返回值类型	公开	备注
_加入学生按钮_被单击			

变量名	类型	静态	数组	备注
新学生	学生			

新学生.加入学生 (1, “张三”, 100, 99)

新学生.置语文成绩 (95)

加入成员 (学生集合, 新学生)



### 18.1.4 派生类和继承性

类允许派生，可以由一个类派生出多个类，派生出的类继承了其父类的所有方法。例如，程序中有一个类 A，类 B 是类 A 的派生类，那么类 B 将继承类 A 的所有方法。

例如，从上面例子中的“学生”类派生出“转校学生”类。在程序中新建一个名为“转校学生”的类，将其基类设置为“学生”。

加入和“学生”类相同的私有成员，并再加入一个“转校时间”成员；加入一个新方法“置转校时间（）”和“取转校时间（）”，代码入下：

类名	基类	备注	
转校学生	学生		
私有成员名	类型	数组	备注
学号	整数型		
姓名	文本型		
语文成绩	整数型		
数学成绩	整数型		
转校时间	日期时间型		

方法名	返回值类型	公开	备注		
置转校时间		✓			
参数名	类型	参考	可空	数组	备注
设置的时间时间	日期时间型				

转校时间 = 设置的时间时间

方法名	返回值类型	公开	备注
取转校时间	日期时间型	✓	

返回（转校时间）

由于“转校学生”是“学生”的派生类，所以它继承了其父类“学生”的所有方法，下面就生成了一个基于“转校学生”的对象，并调用其继承的方法和自身的方法。代码如下：

变量名	类型	静态	数组	备注
转校学生	转校学生			

转校学生.加入学生 (2, “李四”, 88, 89)

转校学生.置转校时间 ([2005年7月1日])

转校学生.

◆ 加入学生
◆ 取语文成绩
◆ 置语文成绩
◆ 置转校时间
◆ 取转校时间





上述程序中，对象输入后智能语法提示列出了其继承的方法和自身的方法。

### 18.1.5 子类中直接调用父类方法

派生类不但继承了其父类的所有方法，而且可以在子类中直接调用其父类的方法。例如，在“学生”类中新加入一个方法“取数学成绩（）”，代码如下：

类名	基类	备注	
学生			
私有成员名	类型	数组	备注
学号	整数型		
姓名	文本型		
语文成绩	整数型		
数学成绩	整数型		

方法名	返回值类型	公开	备注
取数学成绩	整数型		

返回（数学成绩）

然后在其子类“转校学生”中新建一个“取平均成绩（）”方法，在该方法中直接调用其父类的“取数学成绩（）”和“取语文成绩（）”方法，代码如下：

类名	基类	备注	
转校学生	学生		
私有成员名	类型	数组	备注
学号	整数型		
姓名	文本型		
语文成绩	整数型		
数学成绩	整数型		
转校时间	日期时间型		

方法名	返回值类型	公开	备注
取平均成绩	整数型	✓	

返回（（学生.取语文成绩（）+ 学生.取数学成绩（））÷ 2）

“转校学生”类中可以直接调用其父类的方法，但这里要十分注意一个问题，调用的“取语文成绩（）”和“取数学成绩（）”方法，只是调用了父类的方法而已，并没有调用其父类的私有成员，所以取出的数学和语文成绩是“转校学生”类中“数学成绩”和“语文成绩”。如果子类中调用其父类的“置语文成绩（）”方法，修改的成绩也是子类中的语文成绩，而不是父类的语文成绩。因此在子类中不可以取得和修改其父类的私有成员。



概念之一,在面向对象编程中起着举足轻重

在子类方法重新定义父类方法。子类重新定

多态性, 首先新建易程序并新建一个名为

注

注

方法被调用会返回不同的文本。然后再创

注

格式和其父类的“显示 1 ()”方法完全相





子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
基	基类			
派生	派生类			

```
编辑框1.内容 = 基.显示1 () + #换行符
编辑框1.内容 = 编辑框1.内容 + 基.显示2 () + #换行符
基 = 派生
编辑框1.内容 = 编辑框1.内容 + 基.显示1 () + #换行符
编辑框1.内容 = 编辑框1.内容 + 基.显示2 ()
```

程序中，首先调用基类的“显示1 ()”和“显示2 ()”方法，将返回值显示在编辑框中，然后将子类对象变量赋值给父类变量，这个过程体现出了对象变量的多态性；然后再次调用基类的“显示1 ()”和“显示2 ()”方法，将返回值显示在编辑框中，运行效果如图 18-3 所示：

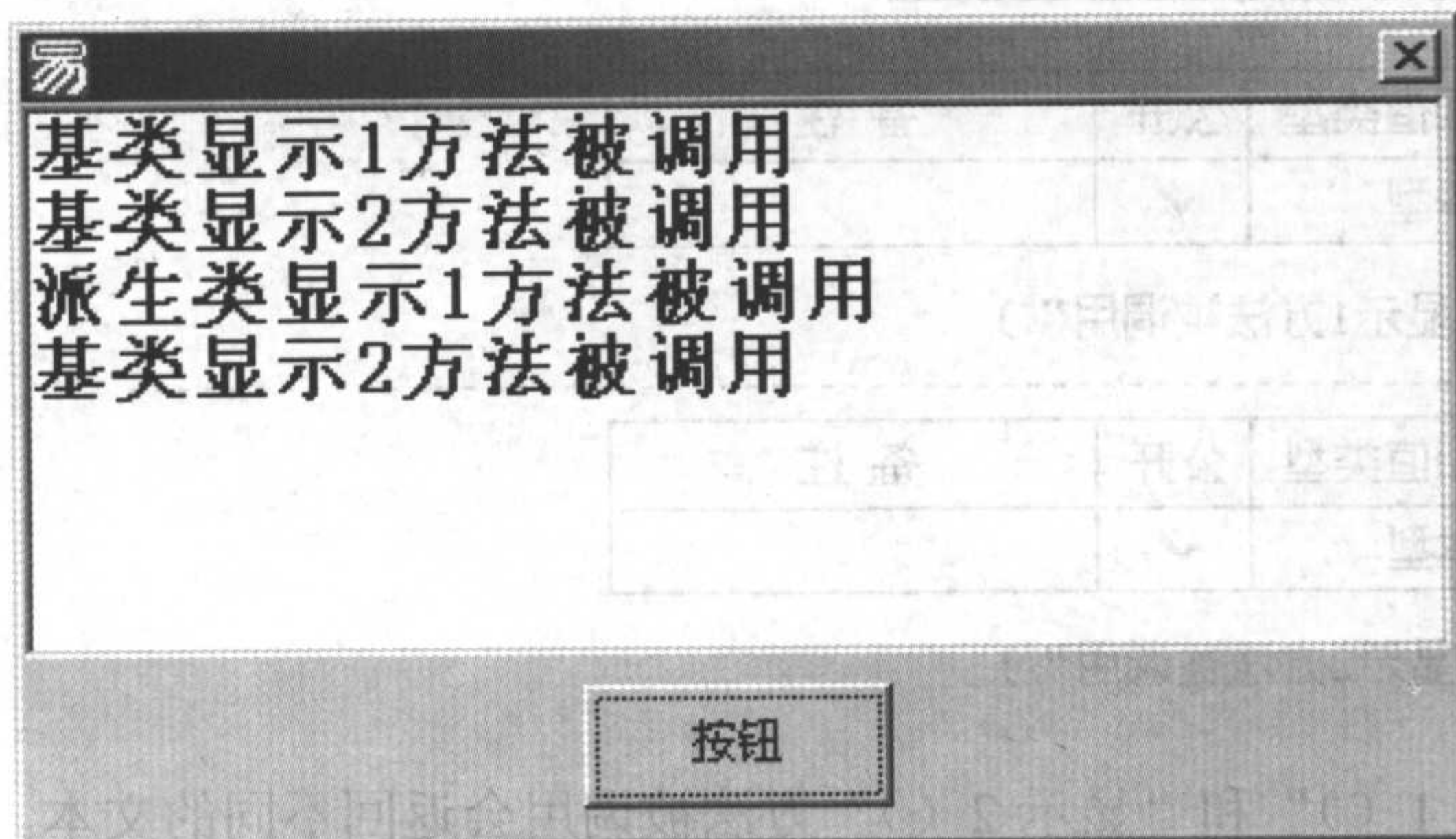


图 18-3 程序运行效果

从显示结果可以清楚地看到覆盖的效果，当子类重定义了父类的“显示1 ()”方法后，运行的是子类“显示1 ()”方法中的代码。

这个例子只是演示了覆盖的效果和对象变量的多态性，可能并看不出多态性有多大的作用。这种覆盖可以用于：父类的某方法提供了一种默认执行的代码，但派生类可能完全覆盖这个方法（派生类的同名方法实现不同的功能）。

但是要体现出多态更大的用途，就需要通过抽象虚拟方法，C++中称为纯虚函数，这种方法只规定命令的格式，而并不编写该方法的实现过程（不写任何代码），该方法如何实现完全通过子类的覆盖。在易语言中，抽象虚方法不用特别声明，只要新建一个空方法，规定方法的参数和属性即可。

下面就举个简单的例子，来了解功能强大的抽象虚方法。首先新建一个名为“计算”的类，定义一个抽象虚方法“计算 ()”，代码如下：



类名	基类	备注
计算		

方法名	返回值类型	公开	备注		
计算		✓			
参数名	类型	参考	可空	数组	备注
计算数值1	整数型				
计算数值2	整数型				
返回计算结果	整数型	✓			

“计算”类中，“计算（）”方法中没有填写任何代码，只规定了这个方法有3个参数，所以这个方法是个抽象虚方法。

然后新建“计算”类的2个派生类“加法”和“减法”。定义和父类相同的私有成员并定义一个“计算”方法，该方法的格式和父类的“计算”方法相同，在方法中编写需要进行的计算，代码如下：

类名	基类	备注
加法	计算	

方法名	返回值类型	公开	备注		
计算		✓			
参数名	类型	参考	可空	数组	备注
加数	整数型				
被加数	整数型				
返回计算结果	整数型	✓			

返回计算结果 = 加数 + 被加数

类名	基类	备注
减法	计算	

方法名	返回值类型	公开	备注		
计算		✓			
参数名	类型	参考	可空	数组	备注
减数	整数型				
被减数	整数型				
返回计算结果	整数型	✓			

返回计算结果 = 减数 - 被减数

“加法”和“减法”类中，只有计算方法，用来实现加法和减法运算，下面，在窗口中添加1个按钮和1个编辑框组件，然后在“\_按钮1\_被单击”子程序中编写代码：





子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
加法对象	加法			
减法对象	减法			
计算结果	整数型			

```
加法对象.计算 (100, 200, 计算结果)
编辑框1.内容 = 到文本 (计算结果) + #换行符
减法对象.计算 (100, 200, 计算结果)
编辑框1.内容 = 编辑框1.内容 + 到文本 (计算结果)
```

程序中，首先在没有使用对象变量赋值的方法时，观察子类如何将父类方法覆盖。运行程序，查看运行效果，如图 18-4 所示。

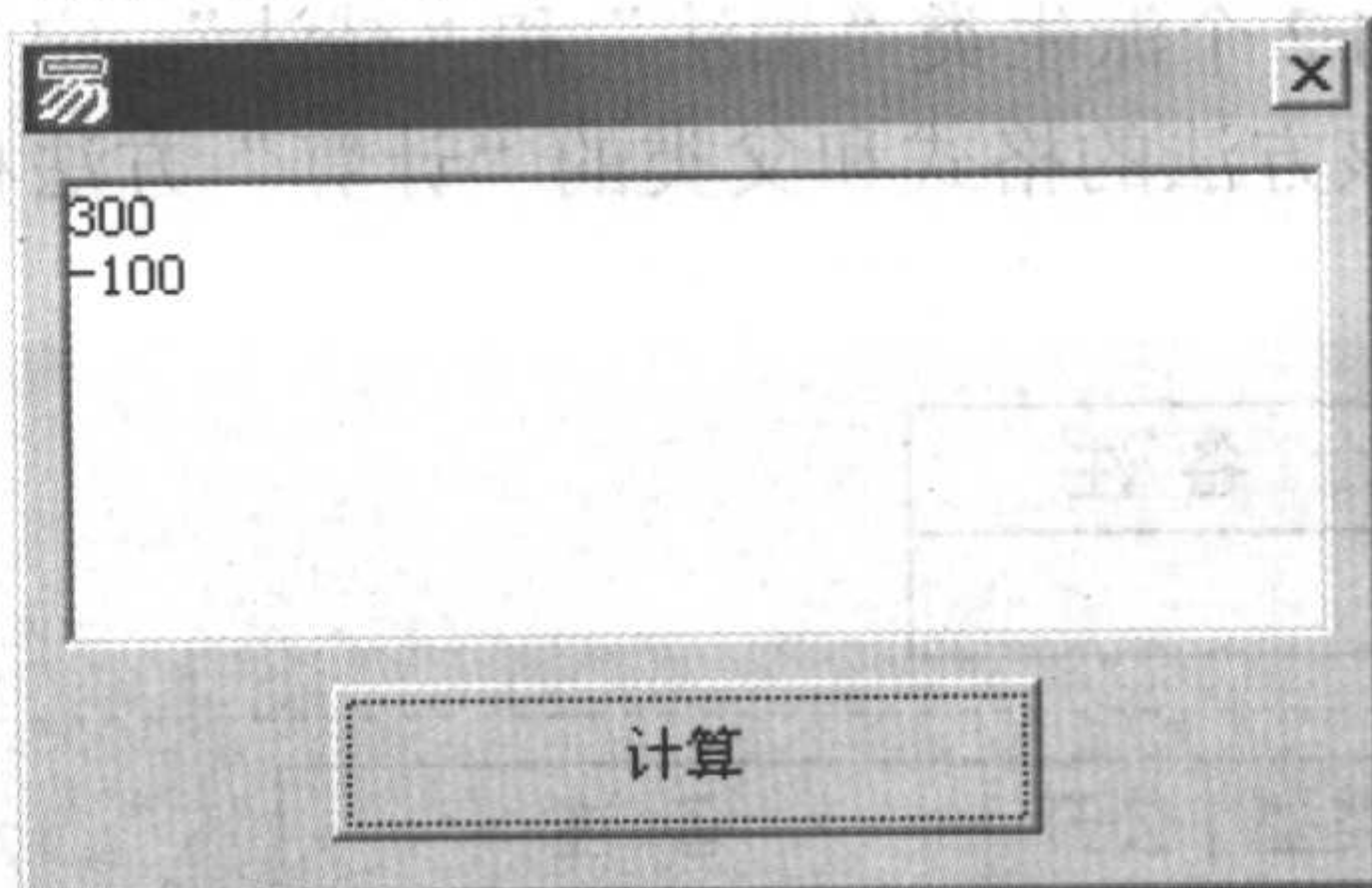


图 18-4 显示计算结果

从显示结果可以看出，当“加法对象”中的“计算 ()”方法被调用，则“加法”类中的“计算 ()”方法将其父类的“计算 ()”方法覆盖，程序执行加的运算；当“减法对象”的“计算 ()”方法被调用，则程序执行减法运算。

程序也可以使用对象变量赋值的方法来达到覆盖的目的，使“计算”类的“计算 ()”方法也具有加减功能，代码如下：

子程序名	返回值类型	公开	备注
按钮2_被单击			

变量名	类型	静态	数组	备注
计算对象	计算			
加法对象	加法			
减法对象	减法			
计算结果	整数型			

```
计算对象 = 加法对象
计算对象.计算 (100, 200, 计算结果)
编辑框1.内容 = 到文本 (计算结果) + #换行符
计算对象 = 减法对象
计算对象.计算 (100, 200, 计算结果)
编辑框1.内容 = 编辑框1.内容 + 到文本 (计算结果)
```



运行程序，查看运行结果，如图 18-5 所示：

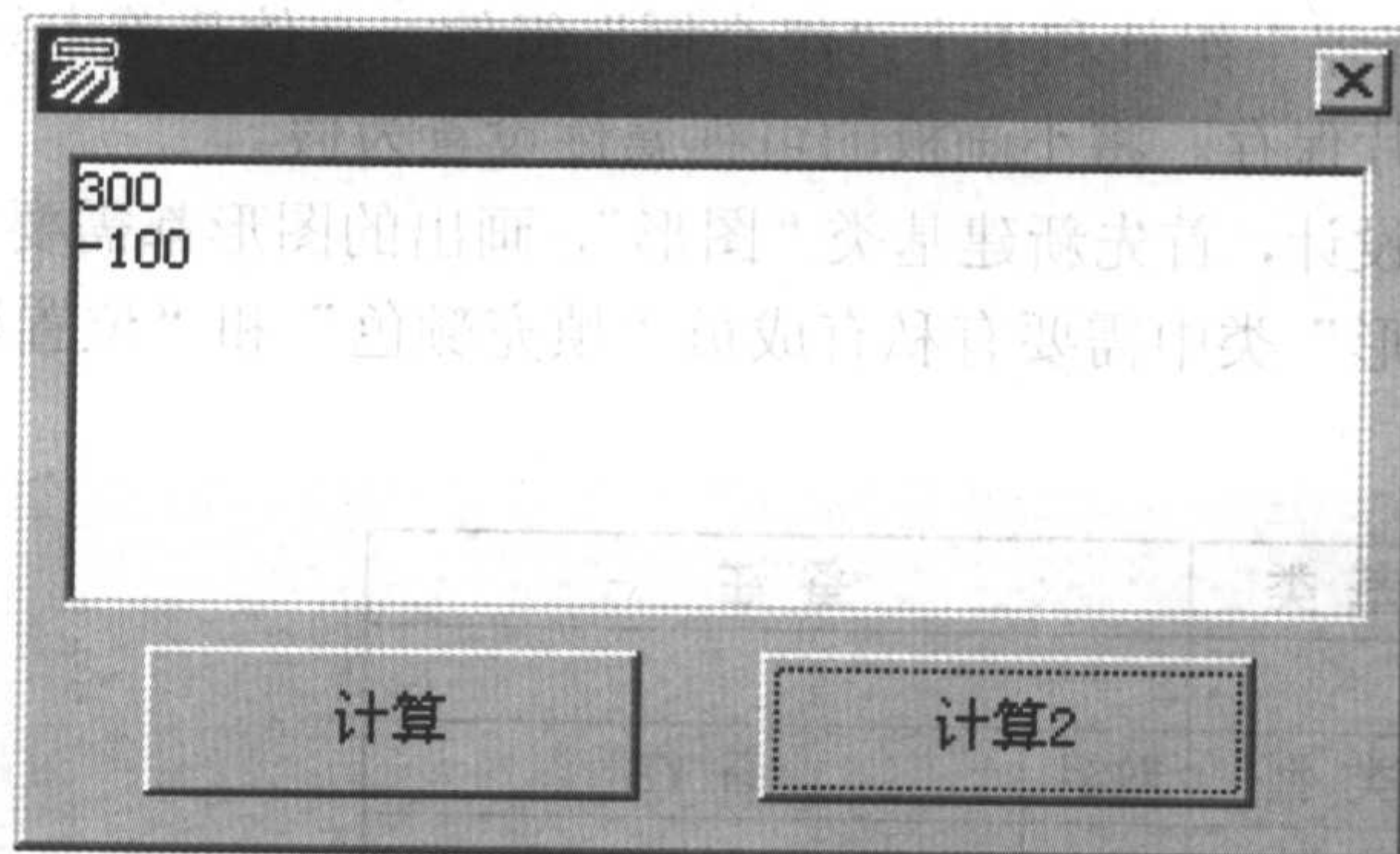


图 18-5 计算后运行效果

第二次的运行结果和第一次是相同的，使用对象变量赋值的方法，同样达到了抽象虚方法的覆盖，在给“计算对象”的赋值不同的子类对象时，“计算对象”的“计算（）”方法执行的运算发生了改变。

以上介绍了关于类的基本知识，大家应该对类有了大概的了解，下面，将结合一个例程，实际应用类的操作和体验一下上面介绍的类的这些特性。

## 18.2 类的实际应用例程

这个例程要实现的功能是：使用者按下鼠标并在画板中拖动，则在画板中画出不同的图形，画板可以画出的图形有“直线”、“矩形”、“椭圆”和“圆角矩形”。

在写本程序前，先考虑这些图形有什么共同特征？都属于什么类型？需要什么共同属性和方法呢？可以确定，这些图形都属于“图形类”，所以大家可以确定了一个基类，就是“图形”，具体的什么图形都是其派生类。考虑到一定需要有一个绘画的方法，用来在画板中画出不同的图形，由于需要画出不同的图形，所以在基类图形中不能确定其派生类需要画什么图形，因此要新建一个抽象虚方法“绘画（）”，该方法只提供一个方法框架，其他需要的方法可以根据需要加入。

下面首先进行程序界面的设计，新建一个易程序，在窗口中添加两个画板组件和工具条组件等，两个画板，其中一个画板用来缓存上一次的画板内容，如图 18-6 所示。

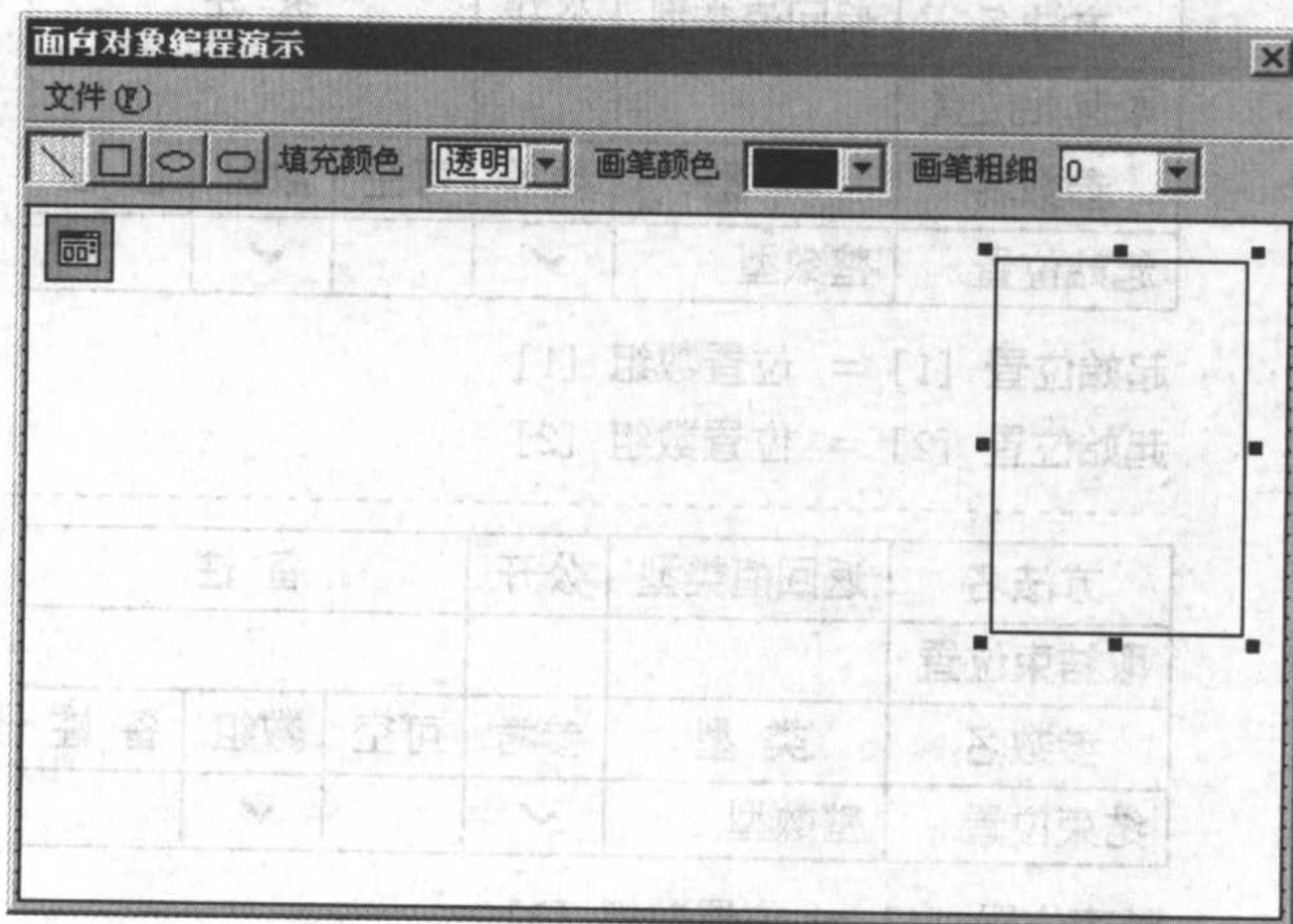


图 18-6 例程界面





工具条中加入了 4 个按钮，按钮类型都是“单选按钮”，在工具条上有 3 个“标签”组件、2 个“颜色选择器”组件和 1 个“组合框”组件。文件菜单中只有一个“保存”选项，用来将画板中的图片保存。将小画板的可视属性设置为假。

下面进行类的设计，首先新建基类“图形”，画出的图形都需要有填充颜色和图形画出的位置，所以“图形”类中需要有私有成员“填充颜色”和“位置数组”。

代码如下：

类名	基类	备注	
图形			
私有成员名	类型	数组	备注
填充颜色	整数型		
位置数组	整数型	4	左边、顶边、右边、底边

方法名	返回值类型	公开	备注
_初始化			当基于本类的对象被创建后，此方法会被自动调用

填充颜色 = 取颜色值 (255, 255, 255)  
默认填充颜色为白色

方法名	返回值类型	公开	备注
取颜色	整数型		类中的成员数据只能在本类的代码中访问

返回 (填充颜色)

方法名	返回值类型	公开	备注		
置颜色		✓	公开以便可以在类代码外部通过对象来访问本方法		
参数名	类型	参考	可空	数组	备注
颜色	整数型				

填充颜色 = 颜色

方法名	返回值类型	公开	备注		
取起始位置					
参数名	类型	参考	可空	数组	备注
起始位置	整数型	✓		✓	

起始位置 [1] = 位置数组 [1]

起始位置 [2] = 位置数组 [2]

方法名	返回值类型	公开	备注		
取结束位置					
参数名	类型	参考	可空	数组	备注
结束位置	整数型	✓		✓	

结束位置 [1] = 位置数组 [3]

结束位置 [2] = 位置数组 [4]



方法名	返回值类型	公开	备注		
置起始位置		✓			
参数名	类型	参考	可空	数组	备注
左边	整数型				
顶边	整数型				

位置数组 [1] = 左边

位置数组 [2] = 顶边

方法名	返回值类型	公开	备注		
置结束位置		✓			
参数名	类型	参考	可空	数组	备注
右边	整数型				
底边	整数型				

位置数组 [3] = 右边

位置数组 [4] = 底边

方法名	返回值类型	公开	备注		
绘画		✓	本方法的具体实现在继承类中完成		
参数名	类型	参考	可空	数组	备注
图形画板	画板				

上述类中，在“\_初始化”子程序中设置默认填充颜色为白色。“取颜色（）”和“置颜色（）”方法用来取的和设置“填充颜色”。“取起始位置（）”、“取结束位置（）”、“置起始位置（）”、“置结束位置（）”方法用来提供给程序或子类取得和更改起始和结束位置。最后类中定义了一个抽象虚函数“绘画”。

然后，派生出它的子类“直线”、“椭圆”、“矩形”以及“圆角矩形”。“椭圆”类的代码如下：

类名	基类	备注	
椭圆	图形		
私有成员名	类型	数组	备注
起始位置	整数型	2	
结束位置	整数型	2	

方法名	返回值类型	公开	备注		
绘画		✓	覆盖基础类中的同名方法来完成具体绘画操作		
参数名	类型	参考	可空	数组	备注
图形画板	画板				





图形画板.刷子颜色 = 图形.取颜色 ()

' 直接调用基类中的方法获取填充颜色

图形画板.画笔颜色 = 画笔颜色

图形画板.画笔粗细 = 画笔粗细

图形.取起始位置 (起始位置)

图形.取结束位置 (结束位置)

图形画板.画椭圆 (起始位置 [1], 起始位置 [2], 结束位置 [1], 结束位置 [2])

“椭圆”类中，重新定义了其父类的“绘画 ()”方法，该方法首先调用了其父类的“取颜色 ()”方法，用来重新设置目标画板的刷子颜色，然后使用 2 个全局变量“画笔颜色”、“画笔粗细”重新定义目标画板的“画笔颜色”和“画笔粗细”。接着使用父类的“取起始位置 ()”和“取结束位置 ()”方法，将画出椭圆的起始坐标和结束坐标取出，并存放在“椭圆”类的私有成员“起始位置”和“结束位置”中。最后使用画板的“画椭圆 ()”方法在画板中画出椭圆。

其他派生类的代码基本类似，只是改变了画板的画图方法。这里就不演示其他类中的代码了。

接下来继续编写窗口程序集中的代码，双击启动窗口，在“\_\_启动窗口\_创建完毕”子程序中编写代码：

窗口程序集名	保 留	备 注	
启动窗口程序集			
变量名	类 型	数 组	备 注
图形对象	图形	0	可以保存所有以“图形”为基类的对象
类型	整数型		1:直线、2:矩形、3:椭圆、4:圆角矩形
起始位置	整数型	2	左边、顶边
是否开始绘画	逻辑型		
前一次右边	整数型		
前一次底边	整数型		

子程序名	返回值类型	公开	备 注
__启动窗口_创建完毕			

变量名	类 型	静态	数 组	备 注
直线图形	直线			
矩形图形	矩形			
椭圆图形	椭圆			
圆角矩形	圆角矩形			

' 将不同的图形对象加入到数组

加入成员 (图形对象, 直线图形)

加入成员 (图形对象, 矩形图形)



加入成员 (图形对象, 椭圆图形)  
加入成员 (图形对象, 圆角矩形)  
类型 = 1

程序中, 创建了 6 个程序集变量, 其中“图形对象”变量是一个“图形”类型的数组。启动窗口创建完毕后, 就将其 5 个派生类对象加入“图形”对象数组中, 相当于将子类对象变量赋值给父类对象变量。

然后编写“画笔颜色选择器”颜色被改变事件子程序代码, 和“组合框 1”项目被改变事件子程序代码:

子程序名	返回值类型	公开	备注
_画笔颜色选择器_颜色被改变			

画笔颜色 = 画笔颜色选择器. 颜色

子程序名	返回值类型	公开	备注
_组合框1_列表项被选择			

画笔粗细 = 到数值 (组合框1. 取项目文本 (组合框1. 现行选中项))

当“画笔颜色选择器”的颜色被改变时, 就将改变后的颜色值存放在“画笔颜色”变量中。

“工具条”被单击事件子程序中的代码如下:

子程序名	返回值类型	公开	备 注		
_工具条1_被单击					
参数名	类 型	参考	可空	数组	备 注
按钮索引	整数型				

类型 = 按钮索引 + 1

\_填充颜色选择器\_颜色被改变 ()

当工具条被单击, 就将“类型”变量赋值按钮索引加 1, 由于按钮索引是从 0 开始, 所以赋值的时候加 1。类型变量用来作为调用“图形对象”数组时的成员索引。

下面继续编写“画板 1”鼠标左键被按下和放开事件子程序代码, 当鼠标左键被按下, 则要取出所画出图形的起始点坐标。代码如下:

子程序名	返回值类型	公开	备 注		
_画板1_鼠标左键被按下	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				





捕获鼠标 (画板1.取窗口句柄 ())

图形对象 [类型].置起始位置 (横向位置, 纵向位置)

起始位置 [1] = 横向位置

起始位置 [2] = 纵向位置

画板2.底图 = 画板1.取图片 ( )

是否开始绘画 = 真

上述程序中, 使用到了一个 API 函数“捕获鼠标”, 该 API 用于在鼠标按钮按下时, 这个窗口会为当前应用程序或整个系统接收所有鼠标输入。该函数的原型为“SetCapture”。函数只有一个参数, 为整数型, 存放要捕获鼠标窗口的窗口句柄。

子程序中, 还使用了画板 2 来保存画板 1 中的图片。

当左键被放开则要取出所画出图形的结束点坐标, 并使用“图形对象”的“置结束位置 ()”方法, 将结束点坐标赋值给“图形对象”的私有成员, 最后调用“图形对象”的“绘画 ()”方法, 画出图形, 代码如下:

子程序名	返回值类型	公开	备注		
_画板1_鼠标左键被放开	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

是否开始绘画 = 假

连续赋值 (0, 前一次右边, 前一次底边)

图形对象 [类型].置结束位置 (横向位置, 纵向位置)

图形对象 [类型].绘画 (画板1)

释放鼠标 ()

上述程序的最后使用到了一个 API 函数“释放鼠标”, 该 API 用于为当前的应用程序释放鼠标捕获。API 原形为“ReleaseCapture”, 没有参数。

子程序中还使用到了“对象[x].方法名 ()”的用法, “图形对象”数组中的成员是不同的子类对象变量, 所以调用数组中不同成员的“绘画 ()”方法可以画出不同的图形。

该程序当鼠标在画板中移动时, 也要追踪鼠标并画出图形, 这个图形大小跟随鼠标移动而改变大小, 所以, 要在“画板 1”鼠标位置被移动事件子程序中编写代码:

子程序名	返回值类型	公开	备注		
_画板1_鼠标位置被移动	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

变量名	类型	静态	数组	备注
图片号	整数型			



```

--- 如果真 (是否开始绘画 = 真)
    --- 如果真 (前一次右边 ≠ 0)
        画笔颜色 = 画板1.画板背景色
        图形对象 [类型].置结束位置 (前一次右边, 前一次底边)
        图形对象 [类型].绘画 (画板1)
        画笔颜色 = 画笔颜色选择器.颜色
        图形对象 [类型].置结束位置 (横向位置, 纵向位置)
        图形对象 [类型].绘画 (画板1)
        图片号 = 载入图片 (画板2.底图)
        画板1.画图片 (图片号, 0, 0, 画板1.宽度 - 2, 画板1.高度 - 2, #拷贝)
        卸载图片 (图片号)
        图形对象 [类型].绘画 (画板1)
        前一次右边 = 横向位置
        前一次底边 = 纵向位置

```

当鼠标在画板中移动, 则先用画板 1 的底色, 画出上次鼠标移动时画出的图像, 然后画出鼠标移动的新位置的图像, 并且每次移动都使用画板 1 画出画板 2 保存的图片, 这样就防止了画板 1 画出的图形会将以前的内容擦掉。

程序中“保存”菜单实现保存图片的功能, 代码如下:

子程序名	返回值类型	公开	备注
_保存_被选择			

```

--- 如果真 (通用对话框1.打开 () = 真)
    --- 如果 (写到文件 (通用对话框1.文件名, 画板1.取图片 ()))
        信息框 (“保存成功!”, #信息图标, “信息”)
        信息框 (“保存失败!”, #错误图标, “错误”)

```

最后, 运行程序, 按下工具条中不同的按钮, 在画板中画出所选择的图形。如图 18-7 所示:

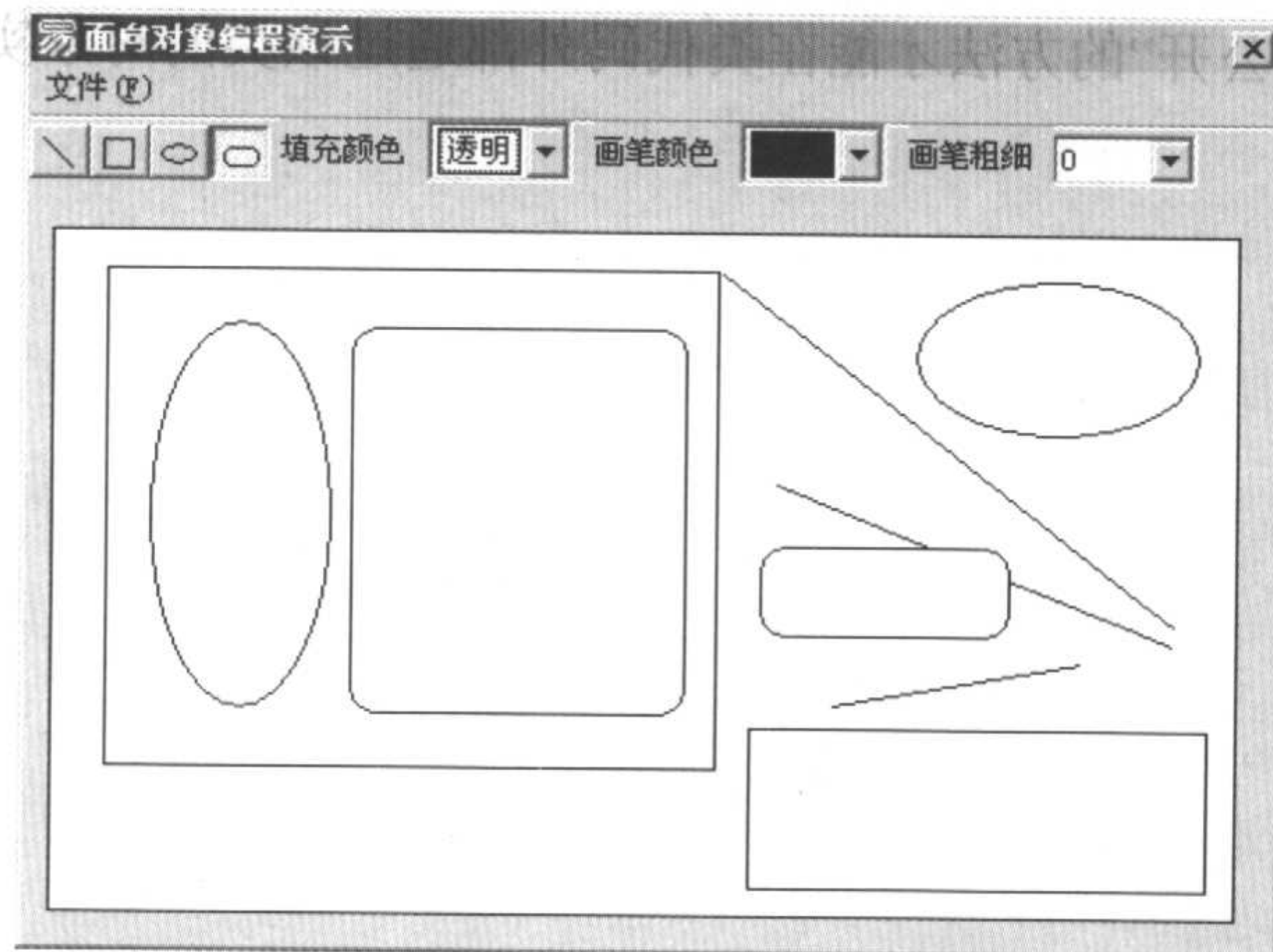


图 18-7 程序运行效果





## 18.3 本章小结

易语言与其他编程语言一样，支持当今先进的编程理念，如面向对象的程序设计方法、面向事件的消息处理机制等等，了解、学习易语言对掌握其他编程语言具有桥梁的作用。

易语言支持用户定义和使用对象，支持类的构造、析构、继承、虚拟方法、多态、封装等特性。

下面就对具体的定义了解一下：

### 1. 对象的构造

构造顺序为：先构造基类对象，再构造其继承类对象，如果类中具有对象成员，则先于其所处对象构造。

### 2. 对象的析构

析构顺序为：先析构继承类对象，再析构基类对象，如果类中具有对象成员，则在其所处对象后析构。

### 3. 继承

任何类均可以指定另外一类作为其基类，继承层数不限。

### 4. 虚拟方法

在基类中的方法可以被其继承类中的同名方法覆盖，当调用此方法时，系统自动根据所调用的对象实体去调用对应的方法。

### 5. 多态性

可以将一个继承类对象赋予到其基类数据类型变量中，此时对此基类对象变量进行操作，将反映出继承类对象的特性。

### 6. 类的封装

● 类的所有成员数据变量只能由该类本身的方法代码所访问，属于私有性质。

● 在继承类中可以以“类名.方法名”的方式指定访问基础类中的方法。

● 只有标记为“公开”的方法才能在类代码外部通过该类的对象实体来访问。



## 第十九章 Linux 程序编写

### 19.1 Linux 简介

Linux 是一个允许在 GNU 公共许可权限下免费使用和自由传播的操作系统。该系统由芬兰人 Linus Torvalds 编写，与 Unix 操作系统兼容，具有 Unix 操作系统的全部功能。

在国内，早在三四年前，一些高校和科研所就开始把 Linux 作为科学与工程计算平台使用。现已有多种中文版 Linux 平台，如 Xteam Linux、中文 Turbo Linux、红旗 Linux、蓝点 Linux 等。鉴于 Linux 高度安全性和公开源代码，各国政府也倾向于推广应用 Linux 操作系统以保障国家安全。目前 Linux 环境下的应用程序数量相对较少，可操作性方面尚不如 Windows 操作系统，但随着技术的进步，Linux 很有可能会取代 Windows 的地位。

易语言从 3.6 版开始，增加了对 Linux 平台的支持。使用易语言，可以很方便地编写 Linux 控制台程序。如使用网络通讯支持库，可以方便的实现网络通讯功能；加上多线程支持库，可以实现多用户并发服务端程序的开发；特别是 MYSQL 支持库封装了丰富的 MYSQL 数据库控制命令，可以直接对 MYSQL 数据库进行操作，极大的方便了 Linux 环境下数据服务系统的开发工作。

### 19.2 创建 Linux 程序

易语言所特有的交叉跨平台功能，让易语言用户可以在 Windows 平台下开发 Linux 环境下的应用程序，Linux 命令的语法与使用方法同对应的 Windows 命令是一样的。

选择菜单“程序”→“新建”，在弹出的“新建”对话框中选择“Linux 控制台程序”，单击“确定”，可以创建在 Linux 环境下运行的控制台程序。如图 19-1 所示。

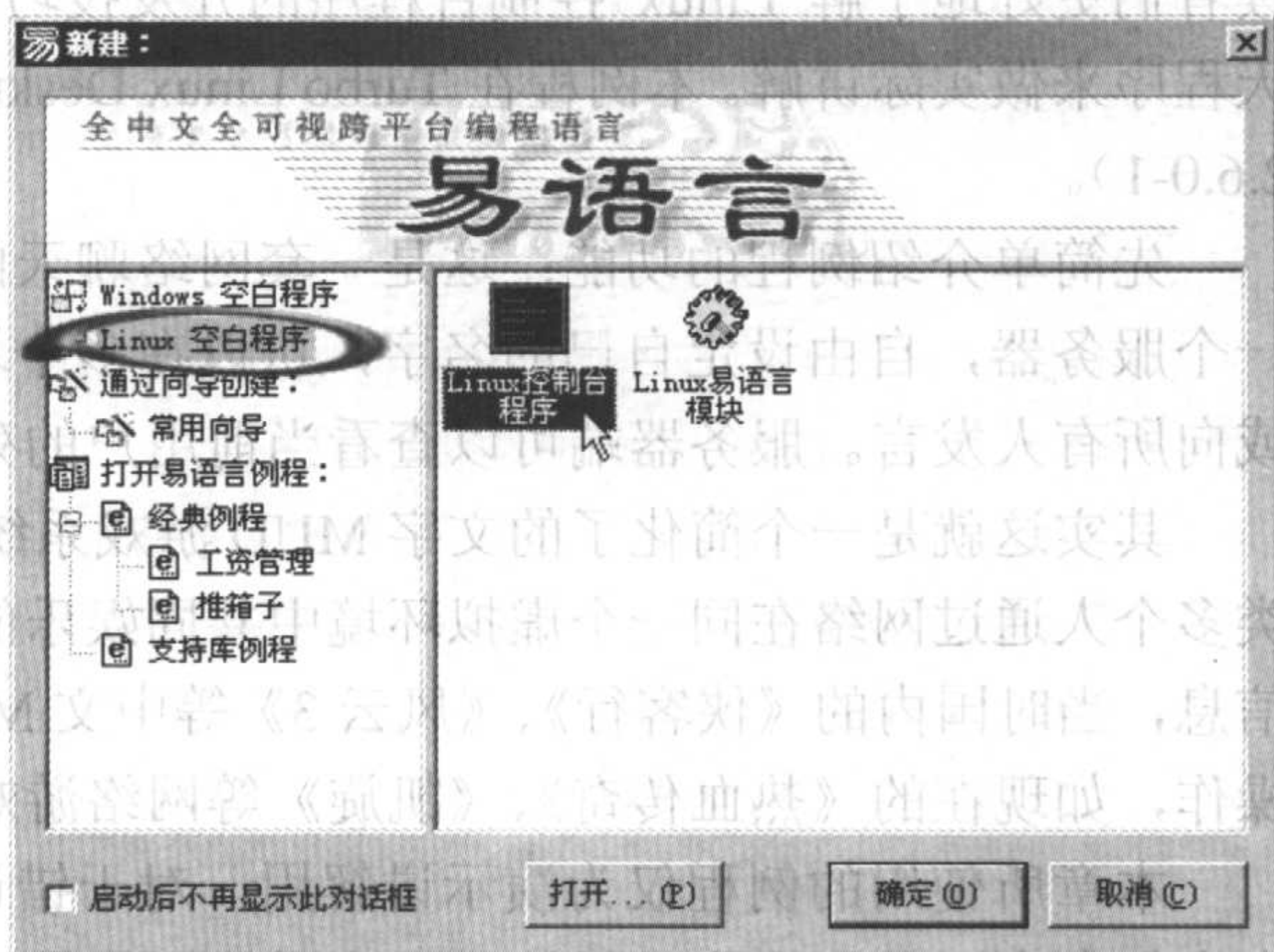


图 19-1 新建一个“Linux 控制台程序”





从支持库面板中可以看到,易语言目前附带的标准 Linux 支持库有系统核心支持库、MySQL 支持库、网络通讯支持库、文字编码转换支持库、数据操作支持库一、多线程支持库、XML 解析支持库、数据结构支持库、数值计算支持库等。如图 19-2 所示。

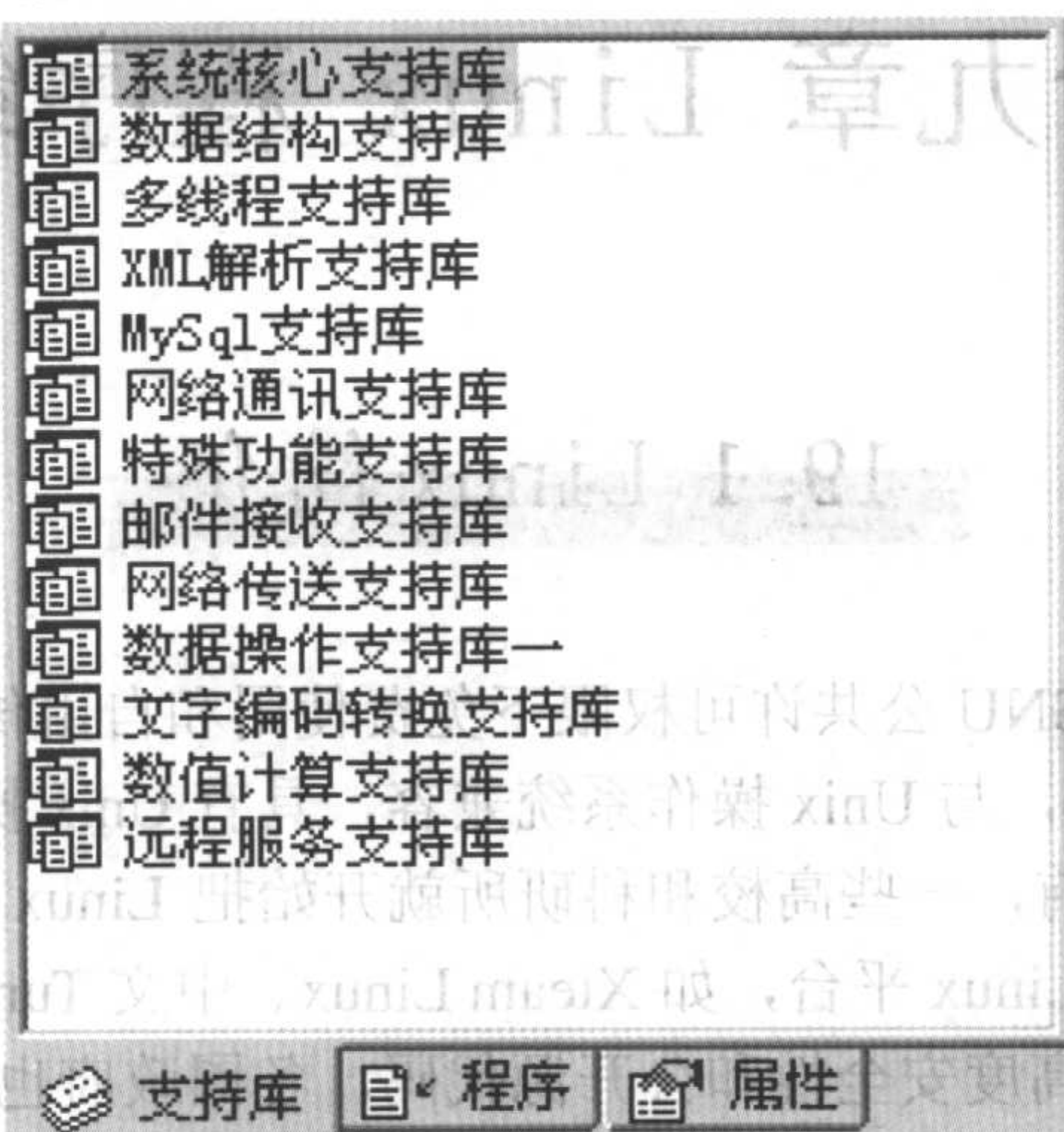


图 19-2 易语言标准 Linux 环境支持库

目前,易语言编写 Linux 环境下的控制台程序,信息与指令的输出、输入必须使用“标准输出”和“标准输入”两条命令。随着易语言功能的不断完善与扩充,现有支持库的功能也会不断增强,以后将支持开发 Linux 环境下的窗口应用程序。

### 19.3 Linux 程序开发与运行

在了解 Linux 操作平台的一些命令之后,就可以使用易语言开发 Linux 程序了。为了让读者们更好地了解 Linux 控制台程序的开发技巧,本节将通过一个简化了的多场景网络聊天程序来做实际讲解。本例程在 Turbo Linux Desktop10.0 环境下调试通过,内核版本(Linux 2.6.0-1)。

先简单介绍例程的功能:这是一套网络聊天的程序,每个人通过客户端程序登录到同一个服务器,自由设定自己的名字,可以在多个场景环境中自由移动,选择指定的人对话或向所有人发言。服务器端可以查看当前用户的列表,并可通过指令启动或停止服务。

其实这就是一个简化了的文字 MUD 游戏系统。MUD (Multi User Dungeon) 游戏是一类多个人通过网络在同一个虚拟环境中共同娱乐的游戏。早期 MUD 只能发送和接收文字信息,当时国内的《侠客行》、《风云 3》等中文 MUD 红极一时;后来逐步发展到图形化的操作,如现在的《热血传奇》、《凯旋》等网络游戏,更是吸引了无数人为之痴迷。

本章所使用的例程仅为演示讲解用,对于错误处理、代码优化、数据保存等问题都没有多加考虑。且为简化代码处理,选用的组件是“网络数据报”,而不是“网络服务器”和“网络客户端”,请在理解程序代码时加以留心。



### 19.3.1 了解例程的相关情况

1. 控制台程序无法实现图形化的窗口操作处理，所有的操作必须使用文本指令，因此编程前需要为服务端和客户端程序定义一份指令表。为了实现各指令的多种快捷输入，采取了每个指令代码前后加空格的方法来实现，这样在检索指令时能够变通的使用寻找文本命令实现智能化指令识别。见表 19-1。

表 19-1 指令表

指令名称	指令代码	使用范围
停止服务	“停止 tingzhi TZ stop”	服务端程序专用
显示用户列表	“列表 liebiao LB LIST”	
启动服务	“启动 qidong QD start”	
帮助	“帮助 bangzhu BZ help”	服务端与客户端 共用
退出	“退出 tuichu TC exit quit”	
登录	“登录 denglu DL login”	客户端程序专用
东	“东 dong D east”	
南	“南 nan N south”	
西	“西 xi X west”	
北	“北 bei B north”	
查看	“查看 chakan CK look”	
对话	“对话 duihua DH talk”	
密谈	“密谈 mitan MT call”	
叫喊	“叫喊 jiaohan JH shout”	

2. 接下来大家要正确地理解程序的设计思路。客户端和服务端程序均需要实时的接收网络传来的信息加以显示和处理，同时也要实时的接收键盘指令加以显示和处理，这两类操作均为异步操作，即一项工作做完才能做下一项工作，因此需要为这两个操作开设独立线程。在程序设计时要避免这两个线程互相直接调用的情况发生，可通过全局变量作为判断标志。

3. 客户端尽量不做数据处理，仅仅是简单的发送指令并获取返回信息，所有的数据处理全部放在服务端进行，这样程序的结构相对清晰，也便于程序的修改与升级。

### 19.3.2 例程服务端代码讲解

现在先看看服务端的程序。请打开随书光盘本章目录中的“MUD 服务端.e”。

在“启动程序集”中，有 5 个子程序：“\_启动子程序”、“服务端监听”、“监控键盘信息”、“客户服务”和“启动服务”。这几个是服务端程序的核心部分。

在“辅助程序集”中，有 6 个子程序：“初始化房间”、“取环境信息”、“取走动信息”、“发言”、“分割命令”和“去首尾空”。

“\_启动子程序”在程序运行最初被自动加载，在其中完成房间信息初始化与启动端口





服务工作，同时为“服务端监听”和“监控键盘信息”子程序创建两个独立的线程，然后就一直等待变量“允许退出”为“真”时再退出程序。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
键盘输入	文本型			

初始化房间 ()

服务器状态. 停止服务 = 真

标准输出 ( "启动易MUD (Linux) 服务程序。" + #换行符)

启动服务 ()

启动线程 (@服务端监听, #轮巡间隔)

启动线程 (@监控键盘信息, )

--- 判断循环首 (服务器状态. 允许退出 = 假)

延时 (1000)

--- 判断循环尾 ()

标准输出 ( #换行符 + #换行符 + "服务已退出。" + #换行符 + #换行符)

延时 (500)

返回 (0)

“服务端监听”子程序中每当收到信息的时候，或等待接收数据的时间超过常量“#轮巡间隔”所规定的时间，就跳出监听端口进程，完成三项工作：

1. 检查是否要求停止服务，如果被要求停止服务，就向每个在线用户发送退出消息，并停止端口监听，退出循环。“服务端监听”子程序代码如下：

--- 判断 (服务器状态. 停止服务 = 真)

--- 如果真 (取数组成员数 (用户信息) > 0)

标准输出 ( "正在发送用户退出信息....." )

--- 计次循环首 (取数组成员数 (用户信息), i)

--- 如果真 (用户信息 [i]. 离线 = 假)

服务端. 发送 (用户信息 [i]. 信息, "服务器退出服务", #发送超时)

--- 计次循环尾 ()

标准输出 ( "已通知所有用户退出。清除用户列表。" + #换行符)

清除数组 (用户信息)

服务端. 关闭 ()

标准输出 ( "服务已停止。" + #换行符 + #换行符)

--- 跳出循环 ()



2. 如果未退出服务, 则判断是否接收到了数据。将未加处理的数据和对方网络信息加入到“用户指令”数组中, 分配一个唯一编号, 交给“客户服务”子程序去处理。

**注意:** 如果访问服务器的人数较多, 可能需要为每个用户指令单独开设一个线程, 这样能保证各用户的响应速度, 但也增大了线程阻塞的可能性。本例程中以“用户指令”数组来保存当前指令, 正是为了满足多线程并行工作方式, 保证端口监控线程能够与客户服务线程相互隔离。将本例程改为多线程的方法见下面设为草稿的那一句程序代码:

```

判断 (取字节集长度 (返回数据) > 0)
    服务器状态. 指令编号 = 服务器状态. 指令编号 + 1
    标准输出 ( 到文本 (服务器状态. 指令编号) + “ ” + 信息. 对方IP
               + “.” + 到文本 (信息. 对方端口) + “ ” + 到文本 (返回数
               据) + #换行符)
    临时用户指令. 编号 = 服务器状态. 指令编号
    临时用户指令. 指令行 = 去首尾空 (到文本 (返回数据))
    临时用户指令. 信息 = 信息
    加入成员 (用户指令, 临时用户指令)
-- 客户服务 (服务器状态. 指令编号)
    
```

3. 检查当前在线用户。因为使用“网络数据报”数据类型, 无法有效检测用户离开事件, 对长时间没有动作的用户, 自动认为已经退出。

“监控键盘信息”子程序是为了能实时了解当前服务器的运作状态而设的。前面程序已经定义了五条服务端指令, 逐一实现即可。本子程序的关键在于: 键盘监控与端口监控是两个独立的线程, 不要在键盘监控过程中直接关闭服务端, 这样可能会导致端口监控线程出现异常错误。正确的做法是通过全局变量对端口监控线程传递关闭标志。

下面是“#退出”与“#停止服务”的代码, 可以看出只是简单的把全局变量标志进行了改变, 而没有直接对数据对象进行操作。

```

判断 (寻找文本 (#退出, 服务器状态. 命令行, 1, 真) > 0)
    服务器状态. 停止服务 = 真
    跳出循环 ()
判断 (寻找文本 (#停止服务, 服务器状态. 命令行, 1, 真) > 0)
    如果 (服务器状态. 停止服务)
        标准输出 (“服务已经停止。” + #换行符)
        服务器状态. 停止服务 = 真
    
```

下面的代码实现了在停止端口服务后, 使用指令重新启动端口服务。这里面调用“启动服务”子程序直接对“网络数据报”对象进行操作, 因为此时端口服务线程已经退出, 无法接收全局变量所传递的启动服务标志。





```

判断 (寻找文本 (#启动服务, 服务器状态.命令行, 1, 真) > 0)
    如果 (服务器状态.停止服务)
        启动服务 0
        启动线程 (&服务端监听, #轮巡间隔)
        标准输出 ( "服务正在运行中, 端口号" + 到文本 (服务器状态.
            端口号) + #换行符)

```

显示用户列表与显示帮助代码比较好理解，在此不再赘述。

接下来看看“客户服务”子程序的代码，这是服务端程序最主要，也是可扩展空间最大的部分。

“客户服务”子程序中使用动态创建的“网络数据报”对象，这样就无需占用主网络数据报对象资源。根据当前数据包的 IP 地址搜索在线用户列表，检查用户是否在线并记录其编号。如果当前发来的是“#登录”请求，判断是临时掉线还是刚刚登录，返回不同的应答信息。当新客户登记了自己的名字后，就可以向服务端发送规定的指令，并获得服务端返回的信息。

下面是对接收到客户端发送过来的查看环境、改换房间、发言等指令处理的代码：

```

-- 判断 (寻找文本 (#查看, 参数 [1], 1, 真) > 0)
    发送文本 = 取环境信息 (用户信息 [用户编号].当前房间, 假, 真)
    临时服务端.发送 (用户信息 [用户编号].信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#东, 参数 [1], 1, 真) > 0)
    发送文本 = 取走动信息 (用户编号, "东", 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#南, 参数 [1], 1, 真) > 0)
    发送文本 = 取走动信息 (用户编号, "南", 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#西, 参数 [1], 1, 真) > 0)
    发送文本 = 取走动信息 (用户编号, "西", 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#北, 参数 [1], 1, 真) > 0)
    发送文本 = 取走动信息 (用户编号, "北", 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#对话, 参数 [1], 1, 真) > 0)
    如果 (取数组成员数 (参数) < 3)
        发送文本 = "指令格式错误.语法: 对话 对方名称 对话内容" + #
            换行符 + #换行符
    发送文本 = 发言 (用户编号, 参数 [2], 参数 [3], 真, 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#密谈, 参数 [1], 1, 真) > 0)

```



```

-- 如果 (取数组成员数 (参数) < 3)
    发送文本 = "指令格式错误。语法：密谈 对方名称 密谈内容" + #
    换行符 + #换行符
    发送文本 = 发言 (用户编号, 参数 [2], 参数 [3], 假, 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)
-- 判断 (寻找文本 (#喊叫, 参数 [1], 1, 真) > 0)
    -- 如果 (取数组成员数 (参数) < 2)
        发送文本 = "指令格式错误。语法：叫喊 叫喊内容" + #换行符 +
        #换行符
        发送文本 = 发言 (用户编号, "", 参数 [2], 真, 临时服务端)
    临时服务端.发送 (指令.信息, 发送文本, #发送超时)

```

从上面代码中可以看出，大部分指令都是调用的近似的子程序，如果为每一个指令写一大段代码，是很不便于代码维护的。公共子程序全部在“辅助程序集”中，读者可以自行查看。

**注意：**当用户退出的时候，并不是从列表中将其删除，而是简单的将其信息清空，这主要是基于多线程环境考虑，如果从数组中删除一个成员，必然会影响其他数组成员的序号。如果采取数据库作为数据保存环境，则可以避免这一情况的发生。

### 19.3.3 例程客户端代码讲解

客户端的主体代码框架和服务端代码框架完全一致，只是少了“客户服务”子程序及其公共子程序部分。请打开随书光盘本章例程中的“MUD 客户端.e”。

程序中“\_启动子程序”的主体代码如下：

```

-- 判断循环首 (真)
    标准输出 ( "请输入服务器IP及端口号 (如192.168.0.115:5566) : " )
    接收文本 = 到半角 (到文本 (标准输入 (真)))
    -- 如果真 (寻找文本 (接收文本, ":", 1, 真) ≤ 0)
        标准输出 ( "输入地址无效, 请重新输入。" + #换行符 )
        到循环尾 ()
    文本数组 = 分割文本 (接收文本, ":", )
    客户端状态.信息.对方IP = 文本数组 [1]
    客户端状态.信息.对方端口 = 到数值 (文本数组 [2])
    如果真 (客户端.发送 (客户端状态.信息, "登录", #发送超时) = 假)
        标准输出 ( "发送连接信息失败, 请确认。" + #换行符 )
        到循环尾 ()
    接收文本 = 到文本 (客户端.接收 (#接收超时, 临时信息))
    -- 如果 (取文本左边 (接收文本, 4) = "欢迎")
        标准输出 ( 接收文本 + #换行符 )

```





```

-- 跳出循环 ()
标准输出 ( "连接服务器失败, 请确认服务器地址与端口。" + #换行符)
-- 判断循环尾 ()
启动线程 (@客户端监听, #轮巡间隔)
启动线程 (@接收键盘信息, )

```

可以看出其流程和服务端程序代码流程没有什么太大区别。所有的数据操作全部在服务端完成, 因此在键盘信息监控中, 除了监控“#退出”指令外, 其他的就只是简单的将指令提交至服务器, 并接收返回结果即可。

同样, 此处通过全局变量“命令行”传递数据至网络监控线程, 而未直接调用“网络数据报”对象, 以保证键盘监控线程与网络监控线程相互独立。

子程序名	返回值类型	公开	备注		
客户端监听					
参数名	类型	参考	可空	数组	备注
超时值	整数型				

变量名	类型	静态	数组	备注
返回数据	字节集			
临时信息	对方信息			

```

-- 判断循环首 (真)
-- 如果真 (客户端状态. 停止服务 = 真)
客户端. 发送 (客户端状态. 信息, "退出", #发送超时)
客户端. 关闭 ()
跳出循环 ()
-- 如果真 (客户端状态. 命令行 ≠ "")
客户端. 发送 (客户端状态. 信息, 客户端状态. 命令行, #发送超时)
客户端状态. 命令行 = ""
客户端状态. 命令完成 = 真
返回数据 = 客户端. 接收 (#轮巡间隔, 临时信息)
-- 如果真 (返回数据 = { })
到循环尾 ()
标准输出 ( 到文本 (返回数据))
-- 判断循环尾 ()
客户端状态. 允许退出 = 真

```

在客户端的端口监控线程中, 定期跳出接收数据状态, 检查是否有指令需要发送。惟一的缺点是在键盘输入指令至一半时接收到了数据, 会直接在当前光标所在位置显示出来。此问题可以通过增加一个变量数组作为数据缓冲区来解决, 具体实现代码请读者自行补充。



### 19.3.4 编译与运行

编译服务端程序和客户端程序，分别保存为 MUDS 和 MUDC。

**注意：**编译出来的 Linux 程序是可以没有后缀名的。文件名也可以取为中文名称，但必须所运行的 Linux 系统支持中文才可以运行。编译前可以在 Windows 环境下进行调试，确保代码无误。

将易语言安装目录 Linux 文件夹下的 krnl.so、sock.so 和 EThread.so 三个文件，随同刚编译的两个程序一并拷贝到 Linux 平台的用户主目录“root”下，在终端(Terminal)中运行程序。

**注意：**Linux 环境下文件名区分大小写。并且文件名前需要加“./”代表程序位于当前目录下。运行服务端程序，直接按回车键使用默认端口，正常启动服务。可以输入服务端指令对服务进行维护。界面如图 19-3 所示。

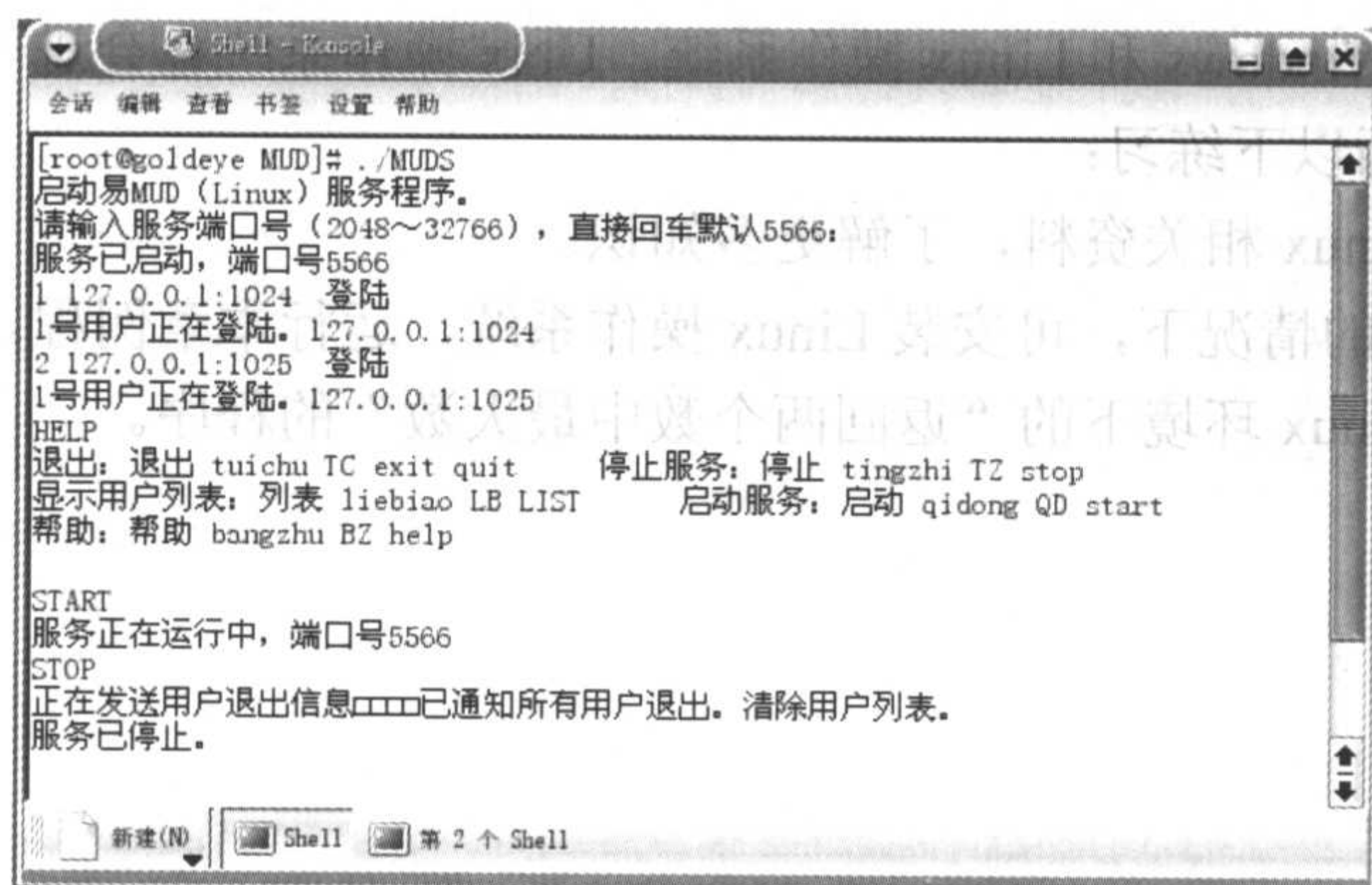


图 19-3 服务端控制界面

运行客户端程序，输入服务器 IP 及端口号登录服务器。定义名字后，即可在几个房间场景中随意走动，如果有几个人同时登录进来，就可以互相对话了。如图 19-4 所示。

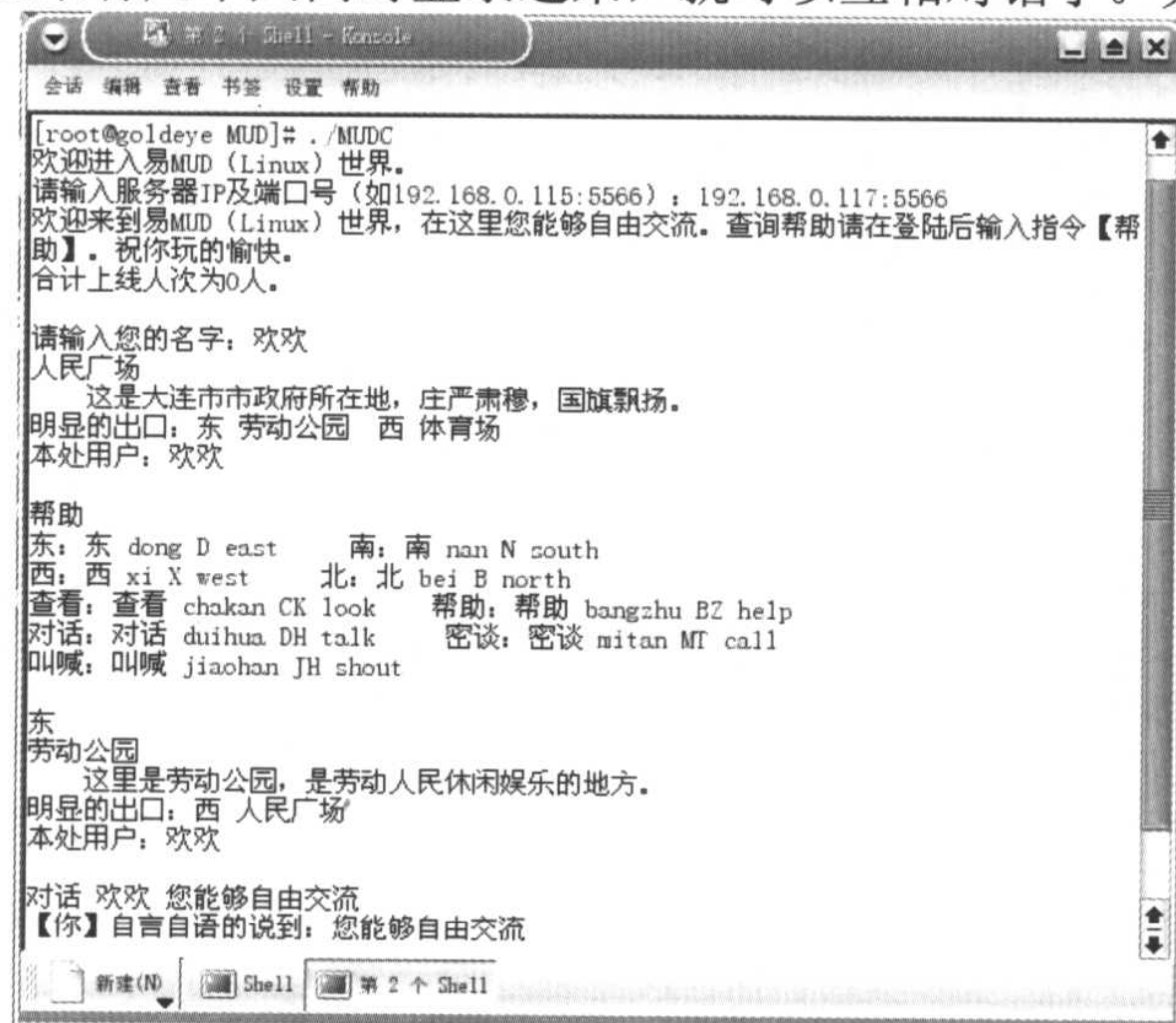


图 19-4 客户端控制界面





**注意：**因为采用“网络数据报”数据类型，因此本例程中通过 IP 是否相同来判断是否同一用户，也就是说，不能在一台电脑上运行两个客户端程序。本例程只能工作在局域网中。

### 19.4 本章小结

易语言跨平台编程无需虚拟机，运行速度快。易语言拥有自己知识产权的编译器，中文源程序被直接编译为目的机器的 CPU 指令，有助于充分发挥目的机器的运行效率。

易语言目的代码与操作系统无关，针对不同操作系统，提供相应的运行时环境。因此易语言不仅可支持 Windows 和 Linux 操作系统，Unix 操作系统也会进一步支持。

大家可自行进行以下练习：

1. 自行查阅 Linux 相关资料，了解更多知识。
2. 在条件允许的情况下，可安装 Linux 操作系统，运行本章例程。
3. 编制一个 Linux 环境下的“返回两个数中最大数”的程序。



## 第二十章 数据结构支持库

易语言的数据结构支持库，是根据数据结构的一些常用算法开发的支持库，用户可以使用该支持库方便地实现例如“链表”、“栈”、“队列”等这些常用的数据结构算法。

### 20.1 数据结构基础

#### 20.1.1 节点

节点是组成一种数据结构的基本单位，后面将介绍的“链表”、“栈”、“队列”等等各种数据结构都要使用到节点。

在易语言的数据结构支持库中，将节点作为单独的对象来操作，若对“链表”、“栈”等等的操作时使用到节点，都要赋值一个节点型的数据。例如：

变量名	类型	静态	数组	备注
链表1	链表			
节点1	节点			

节点1.加入属性 (“数据”, “我爱易语言”)

链表1.加入节点 (节点1, 2)

“节点”对象提供了 10 种方法，用来对节点进行操作，包括“加入属性 ()、删除属性 ()、修改属性 ()、取全部属性名 ()、取属性类型 ()、取数值取逻辑值 ()、取日期值 ()、取文本值 ()、取字节集值 ()”方法。节点必须存在于一种数据结构中，所以这些方法在后面介绍数据结构时会陆续使用到，这里就不多做介绍了。

#### 20.1.2 链表

在前面的章节中，已介绍过易语言的数组变量，大家可以把它看做是一系列连续的数据，数据在计算机的内存中和数组类似，是连续存储的，如果要得到一系列连续存储的数据，就要分配给这些数据足够大的内存空间，并且不容易管理，例如要在这些数据中间插入一个数据，这就需要将插入位置以后的所有数据拷贝到新的一块存储区，插入数据后将移出的数据复制回来，这样就需要提供更大的内存空间。

由于使用这种连续存储数据的方式带来了不便，所以就要寻找一种新的存储结构，以便从连续存储模式下解脱出来，这就是链表出现的原因。





链表是一种数据存储的结构，类似一根链条，使用这种结构存储数据，当插入数据时，需要首先拆除连接，插入一个新链然后重新连接即可；当要删除数据时，只需要拆除链条中需要删除的链，然后重新连接链条即可。如图 20-1 所示。

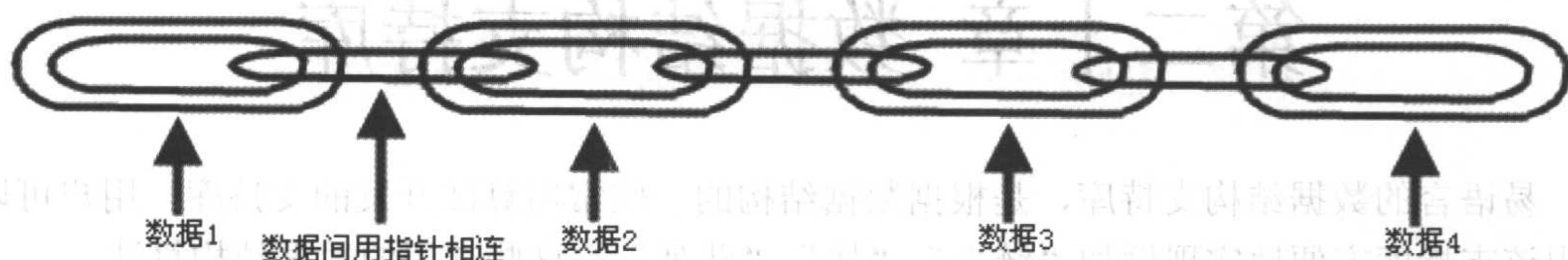


图 20-1 链表结构

上面用了链条来描述“链表”方式存储数据的结构，在实际计算机中使用链表方式存储数据使用的是“节点”（类似链条中的每个链），每个节点由“数据域”和“链接域”（指向链表下一项的指针）组成，数据域用来存放数据，指针是将链表中单个节点维系在一起的纽带。

链表方式存储数据，由于是由单块数据进行存储，然后使用指针将这些数据连接起来，所以不用提供很大的存储空间，并且管理起来十分方便和灵活。如图 20-2 所示。

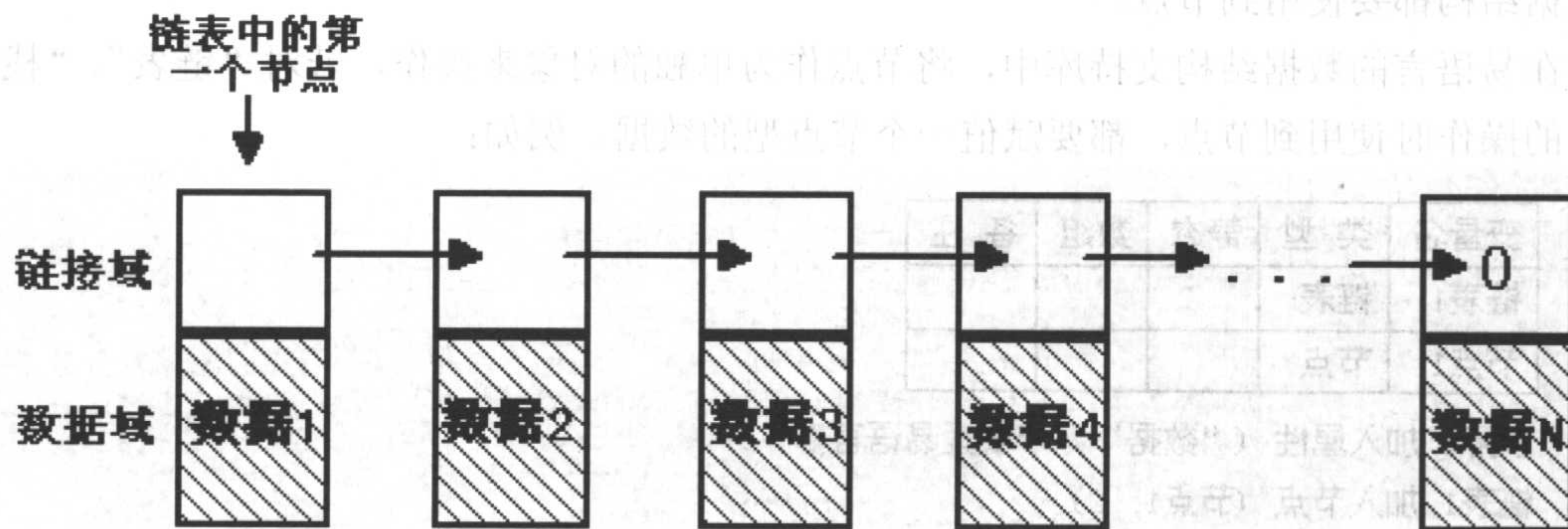


图 20-2 链表描述

由于图 20 - 2 中的每个链表节点都正好有一个链接域，所以该图的链表结构被称之为单向链表（singly linked list）。并且，由于第一个节点的指针指向第二个节点，第二个节点的指针指向第三个节点，最后一个节点链接域为 NULL（或 0），故这种结构也被称作链（chain）。

上面介绍了链表的基本结构，那么这些结构在易语言中如何实现呢？在易语言中可以通过数据结构支持库中的“链表”数据类型实现。该数据类型首先需要创建“链表”对象，可以使用该对象提供的 18 个方法来创建一个链表结构。

链表对象提供的方法包括：“是否为空（）、取大小（）、加入节点（）、删除首节点（）、删除尾节点（）、全部删除（）、删除当前节点（）、删除节点（）、修改节点（）、修改当前节点（）、向下移动（）、向上移动（）、到首节点（）、到尾节点（）、搜索节点（）、取当前节点键值（）、取当前节点（）、取节点（）”。

易语言中链表通过键值来区分节点在链表中的位置（键值类似数组变量中的成员索



引), 向链表中加入和删除节点时, 也要通过键值来判断被插入和删除节点的位置, 键值从 0 开始, 例如:

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
链表1	链表			
节点1	节点			
临时节点	节点			
取出文本	文本型			

节点1. 加入属性 (“数据”, “我爱易语言”)

链表1. 加入节点 (节点1, 0)

节点1. 修改属性 (“数据”, “易语言数据结构支持库”)

链表1. 加入节点 (节点1, 1)

链表1. 到首节点 ()

链表1. 取当前节点 (临时节点)

临时节点. 取文本值 (“数据”, 取出文本)

信息框 (取出文本, 0, )

上述程序代码中, 首先给节点加入一个名为“数据”的属性, 并给该属性赋予文本值“我爱易语言”, 然后向链表中加入该节点, 第二个节点用同样方法加入, 然后用“到首节点 ()”和“取当前节点 ()”方法, 将首节点取出并存放在临时节点中, 然后用节点对象的“取文本值 ()”方法, 将第一个节点存放的文本值取出, 最后用信息框显示该节点的值。

### 20.1.3 栈

栈和队列可能是使用频率最高的数据结构, 二者都来自于线性表数据结构 (经过某种限制以后)。

线性表 (linear list) 是这样的数据对象, 其实例形式为:  $(e_1, e_2, \dots, e_n)$ , 其中  $n$  是有穷自然数。  $e_1$  是表中的元素,  $n$  是表的长度。当  $n = 0$  时, 表为空; 当  $n > 0$  时,  $e_1$  是第一个元素,  $e_n$  是最后一个元素, 可以认为  $e_1$  优先于  $e_2$ ,  $e_2$  优先于  $e_3$ , 如此等等。除了这种优先关系之外, 在线性表中不再有其他结构。

栈数据结构是通过对线性表的插入和删除操作进行限制从而得到的, 插入和删除操作都必须在表的同一端完成。即插入数据是插入到栈的最后, 删除数据是删除栈里的最后一个元素, 因此, 栈是一个后进先出 (last-in-first-out, LIFO) 的数据结构。

堆栈 (stack) 是一个线性表, 其插入和删除操作都在表的同一端进行。其中一端被称为栈顶 (top), 另一端被称为栈底 (bottom)。如图 20-3 所示。



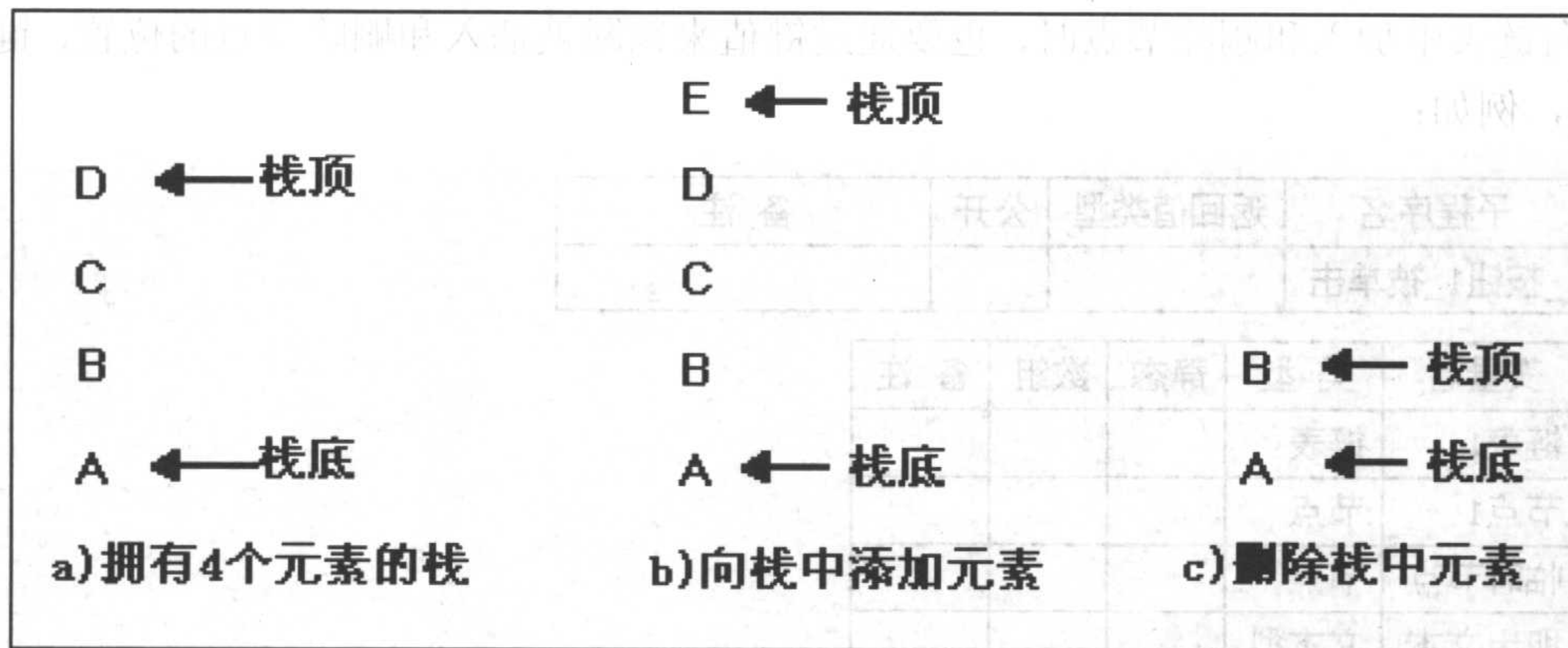


图 20-3 栈的结构

图 20-3a 为一个拥有 4 个元素的栈；图 20-3b 是向栈中添加一个元素 E，则栈顶元素变为 E，可以看到向栈中添加元素是向栈顶添加的；图 20-3c 是删除栈中的 3 个元素，最后添加的元素 E 会被最先删除，然后依次删除栈顶元素，这就是前面说到的后进先出结构。

易语言中，可以通过数据结构支持库中的“栈”对象，来轻松的实现栈的创建和操作，栈中的每一个元素就是一个节点，“栈”对象提供了 6 种方法来操作栈，包括：是否为空、取大小、清空、取栈顶节点、压入、弹出。“压入（）”方法用来向栈中添加元素，“弹出（）”方法用来将栈顶元素取出并删除，只读取栈顶元素而不删除则使用“取栈顶节点（）”方法。例如：

变量名	类型	静态	数组	备注
栈1	栈			
节点1	节点			
数据数组	整数型		10	
循环变量	整数型			

节点1.加入属性（“数据”，0）

--- 计次循环首（10，循环变量）

节点1.修改属性（“数据”，数据数组[循环变量]）

栈1.压入（节点1）

--- 计次循环尾 0

上述程序中使用了计次循环命令将数组“数据数组”中的全部数据添加到“栈1”中。

#### 20.1.4 队列

像栈一样，队列（queue）是一个特殊线性表，其插入和删除操作分别在表的不同端进行。因此，队列是一个先进先出（first-in-first-out, FIFO）的线性表。添加新元素的那一端被称为队尾（rear），而删除元素的那一端被称为队首（front）。如图 20-4 所示。



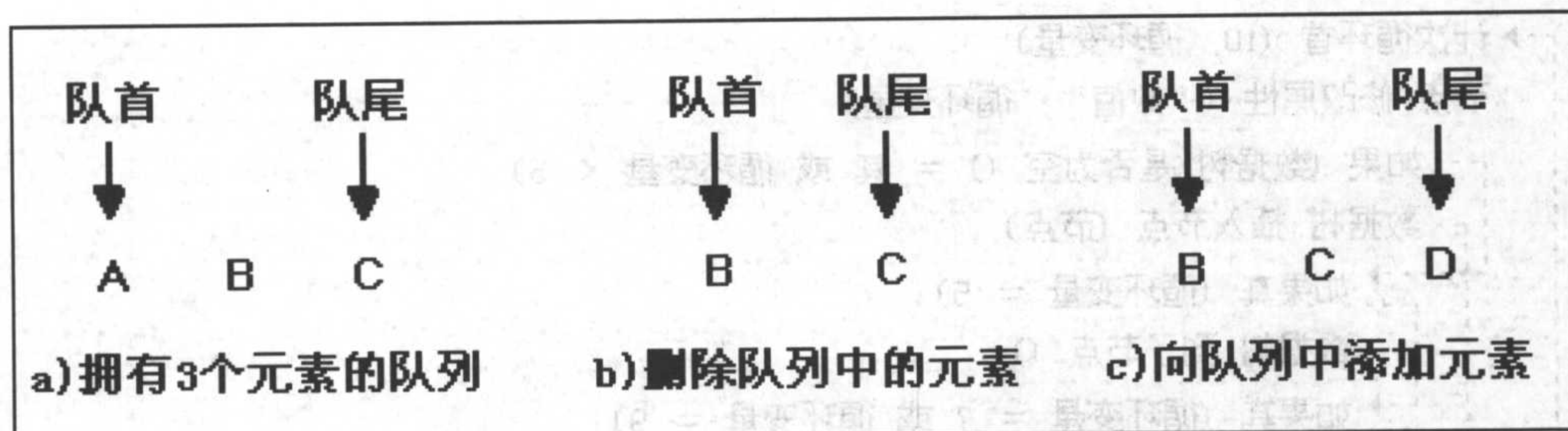


图 20-4 队列结构

图 20-4a 为一个有 3 个元素的队列；图 20-4b 是从队首删除元素 A；图 20-4c 是从队尾向队列中添加新的元素 D。从图中可以看到，最先被添加的元素 A 在删除时首先被删除，体现出了队列先进先出的特点。

易语言中可以通过数据结构支持库中的“队列”对象来创建和操作队列，该对象提供了 6 个方法，包括：“是否为空（）、取大小（）、清空（）、取队列首节点（）、压入（）、弹出（）”。队列的操作和栈类似，区别就在于栈是删除和添加元素都在一端进行，即都在栈顶进行；队列删除和添加元素在两端进行，即删除元素在队首而添加元素在队尾。

### 20.1.5 树

树的结构大家应该并不陌生，在日常生活中和使用计算机时经常可以见到。例如，在一个企业管理层中，假设总裁地位最高，它下属的有销售部副总裁、财务部副总裁、开发部副总裁等等，而这些副总裁又有很多下属，这种有层次的结构就可以看成一种树的结构；在计算机中目录的结构经常被称作目录树，这种结构就是典型的树型结构。

树（tree）是一个非空的有限元素的集合，其中一个元素为根（root），余下的元素（如果有的话）组成它的子树（subtree）。

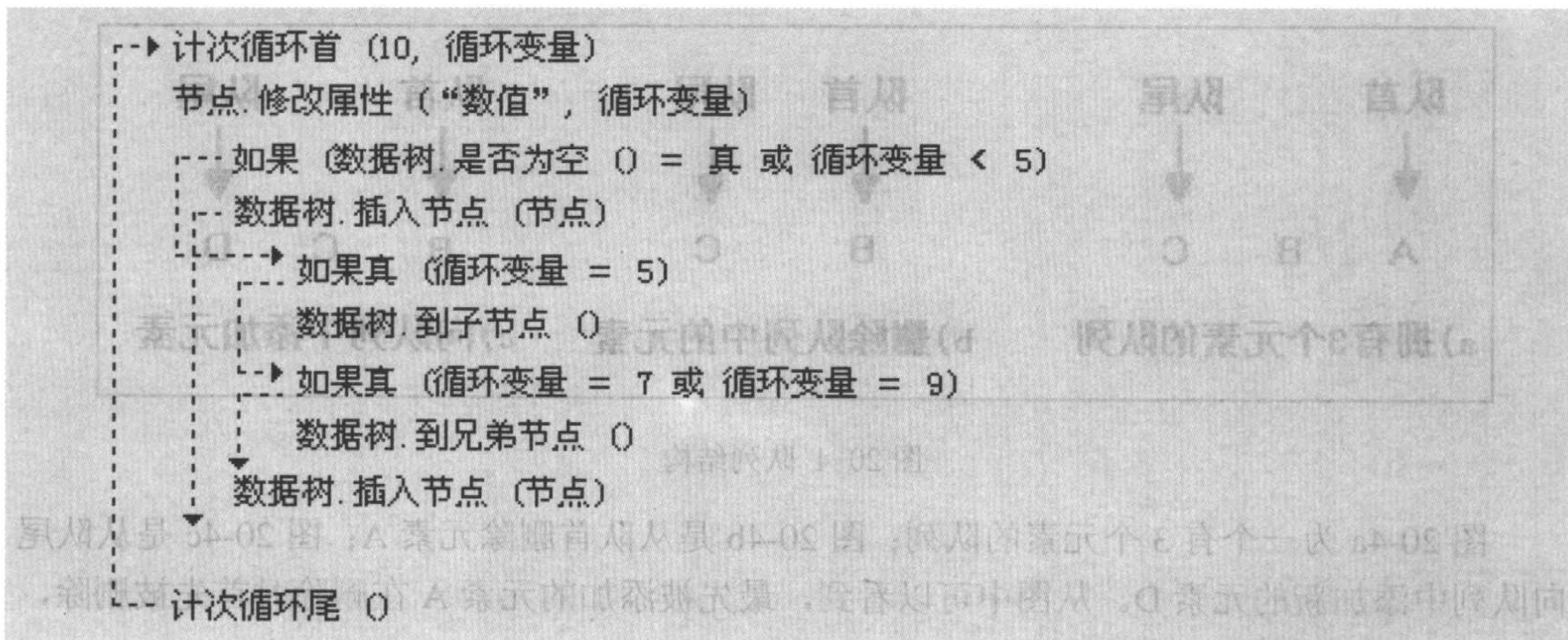
在易语言中，想使数据存储为树的结构，可以使用支持库中“树”对象提供的方法来创建和操作树结构，树结构中的每个元素表示为一个节点，树的操作方法有 12 种，包括：“是否为空（）、取大小（）、清空（）、到子节点（）、到父节点（）、到兄弟节点（）、到根节点（）、取当前节点（）、取子节点数（）、插入节点（）、修改当前节点（）、删除当前节点（）”。

树结构中的根节点为树的根，即层次最高的节点，兄弟节点为和自身同根并同一层次的其他节点。例如编写将 1 到 10 的整数保存到一个树结构中的程序代码如下：

变量名	类型	静态	数组	备注
数据树	树			
节点	节点			
循环变量	整数型			

节点.加入属性（“数值”，0）





代码运行后，1 到 10 的整数就被保存成如图 20-4 所示的树结构中：

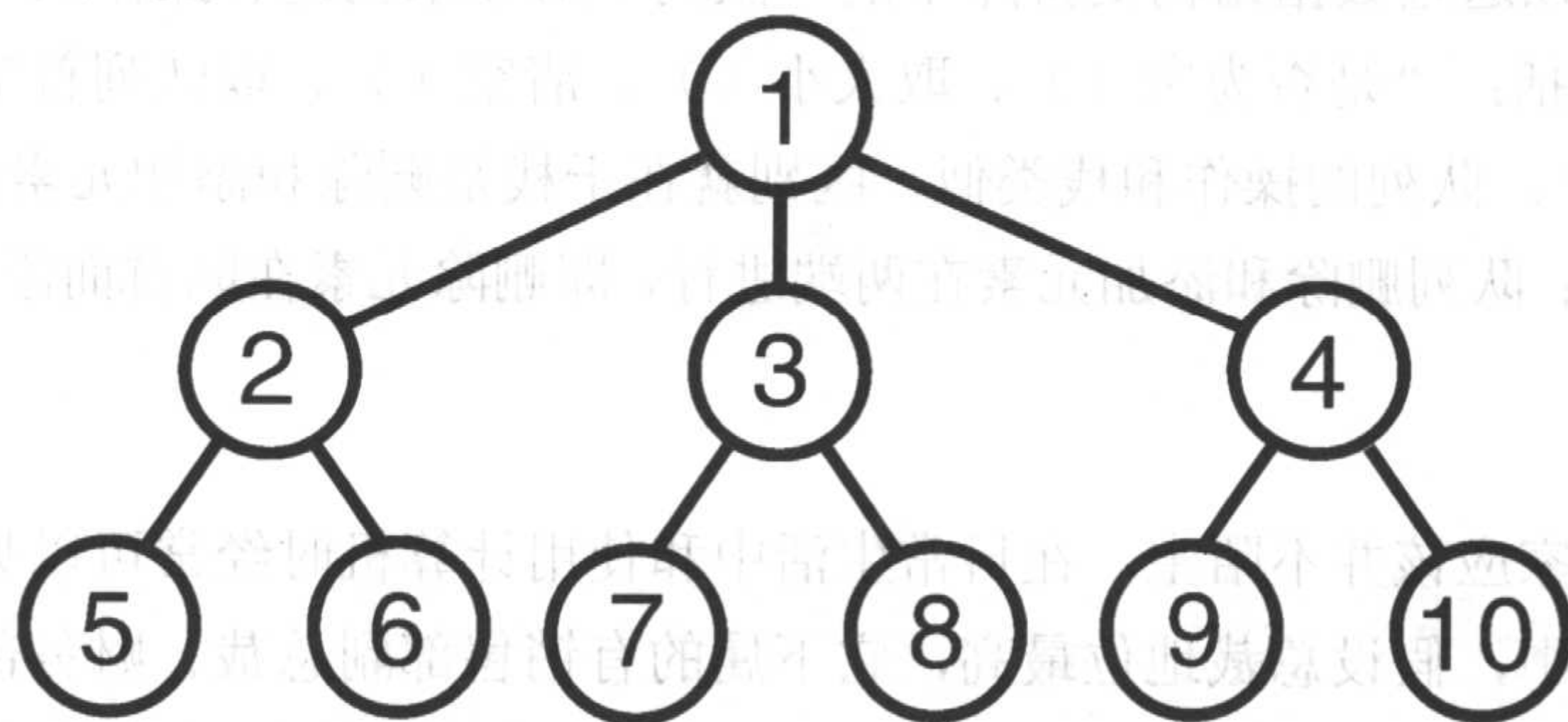


图 20-4 将 1-10 的整数保存成树结构

## 20.1.6 二叉树

二叉树是一种特殊的树结构，它的每个节点最多只能有 2 个子节点。

二叉树（binary tree）是有限个元素的集合（可以为空）。当二叉树非空时，其中有一个称为根的元素，余下的元素（如果有的话）被组成 2 个二叉树，分别称为它的左子树和右子树。

二叉树和树的根本区别是：

1. 二叉树可以为空，但树不能为空。
2. 二叉树中每个元素都恰好有两棵子树（其中一个或两个可能为空）。而树中每个元素可有若干子树。
3. 在二叉树中每个元素的子树都是有序的，也就是说，可以用左、右子树来区别。而树的子树间是无序的。像树一样，二叉树也是根节点在顶部。二叉树左(右)子树中的元素画在根的左(右)下方。



## 20.2 栈的应用例程

讲了这么多的数据结构，那在实际编程时如何使用呢？下面就使用数据结构中的栈，来编写一个简单的计算器。

首先创建一个易程序，添加一个编辑框组件和 1 个按钮组件，按钮组件的标题改为“（”，将按钮名称改为“数字按钮”。由于其他按钮都使用“复制窗口组件”的方法来取得，所以只添加一个按钮组件。如图 20-5 所示。

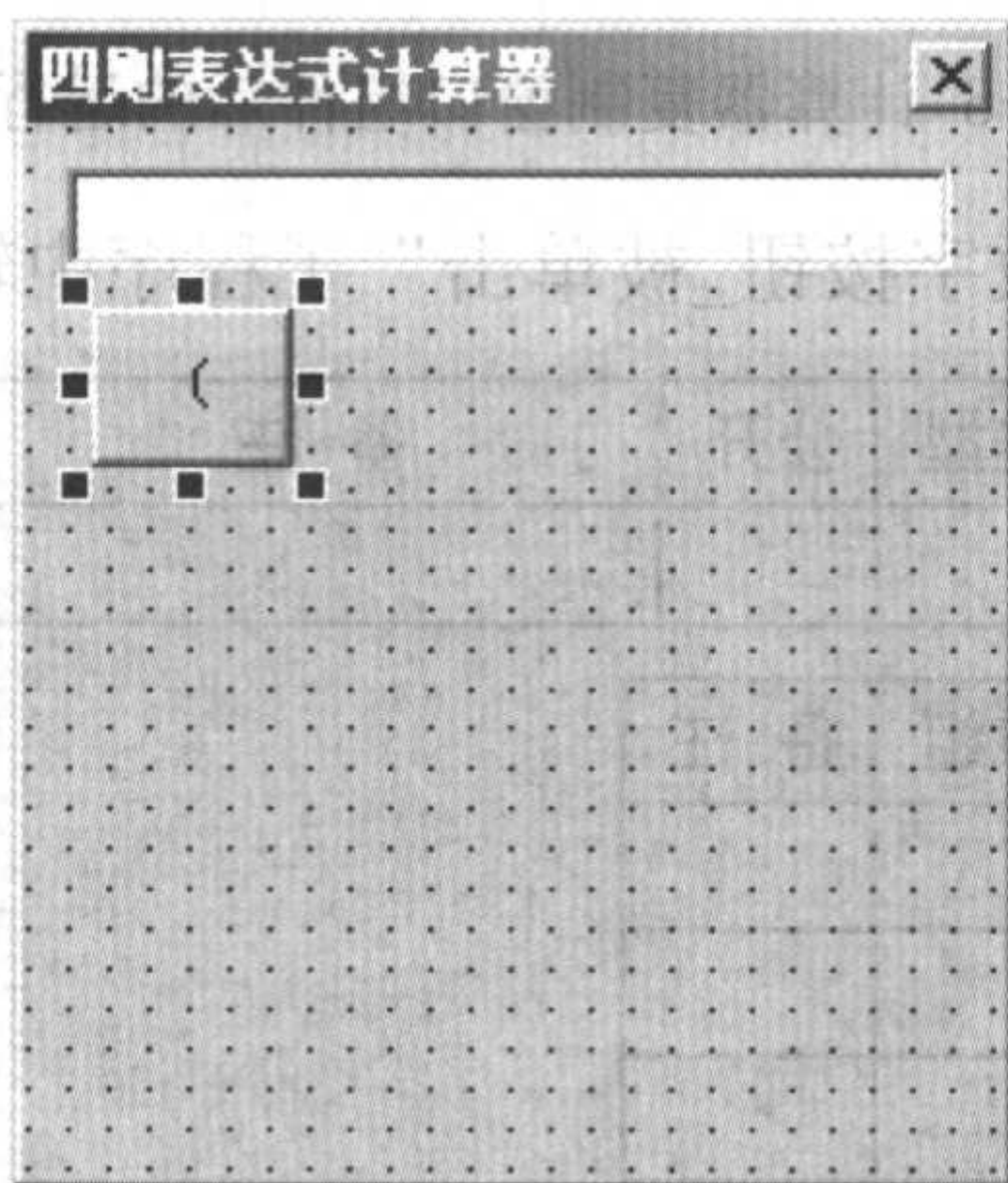


图 20-5 计算器界面

双击窗口，在“\_\_启动窗口\_创建完毕”子程序中输入以下代码：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
变量	整数型			

--- 计次循环首 (19, 变量)

复制窗口组件 (数字按钮, 按键 [变量])

按键 [变量]. 左边 = 数字按钮. 左边 + (变量 % 4) × 43

按键 [变量]. 顶边 = 数字按钮. 顶边 + (变量 \ 4) × 34

按键 [变量]. 标题 = 多项选择 (变量, “)”, “退格”, “清除”, “7”, “8”, “9”, “/”, “4”, “5”, “6”, “\*”, “1”, “2”, “3”, “-”, “0”, “.”, “=”, “+”)

按键 [变量]. 可视 = 真

--- 计次循环尾 0

组合框1. 获取焦点 0

这些代码用来在窗口创建完毕后自动生成计算器中的其他按钮，运行后效果如图 20-6 所示。





图 20-6 计算器复制窗口组件后的运行效果

双击“数字按钮”，在“\_数字按钮\_被单击”子程序中输入以下代码：

子程序名	返回值类型	公开	备注
_数字按钮_被单击			

变量名	类型	静态	数组	备注
临时按钮	按钮			
变量	整数型			
已存在	逻辑型			
文本	文本型			

临时按钮 = 取事件组件 ()

判断 (临时按钮.标题 = “退格”)

编辑框1.内容 = 取文本左边 (编辑框1.内容, 取文本长度 (编辑框1.内容) - 1)

判断 (临时按钮.标题 = “清除”)

编辑框1.内容 = “”

判断 (临时按钮.标题 = “=”)

编辑框1.内容 = 计算表达式 (编辑框1.内容)

编辑框1.内容 = 编辑框1.内容 + 临时按钮.标题

在数字按钮被单击后，判断被按下的按钮并触发相应的事件和执行相应的代码。当“=”号键被按下后，就运行“计算表达式”子程序，“计算表达式”子程序用来对算式进行分析，然后计算并将计算的结果返回。计算表达式子程序代码如下：

窗口程序集名	备注		
窗口程序集1			
变量名	类型	数组	备注
按键	按钮	19	
算符堆栈	栈		
数据堆栈	栈		



子程序名	返回值类型	公开	备注		
计算表达式	文本型				
参数名	类型	参考	可空	数组	备注
表达式	文本型				

变量名	类型	静态	数组	备注
变量	整数型			
数据	文本型			
优先级	文本型			
操作数1	整数型			
操作数2	整数型			
负头	逻辑型			

算符堆栈. 清空 ()

数据堆栈. 清空 ()

表达式 = 子文本替换 (删全部空 (到半角 (表达式)), “(-”, “(0-”, 1, , 假) + “#”

数据 = 取文本左边 (表达式, 1)

负头 = 选择 (数据 = “-”, 真, 假)

--> 判断循环首 (数据 ≠ “#” 或 取算符栈顶 (算符堆栈) ≠ “#”)

如果 ((数据 = “+” 或 数据 = “-” 或 数据 = “\*” 或 数据 = “/” 或 数据 = “(” 或 数据 = “)” 或 数据 = “#”) 且 负头 = 假)

表达式 = 选择 (数据 ≠ “#”, 文本替换 (表达式, 1, 1, ), 表达式)

--> 循环判断首 ()

优先级 = 比较优先级 (取算符栈顶 (算符堆栈), 数据)

-- 判断 (优先级 = “<”)

-- 算符入栈 (算符堆栈, 数据)

--> 判断 (优先级 = “=”)

-- 算符出栈 (算符堆栈)

--> 判断 (优先级 = “>”)

操作数2 = 数据出栈 (数据堆栈)

操作数1 = 数据出栈 (数据堆栈)

-- 数据入栈 (数据堆栈, 计算 (操作数1, 算符出栈 (算符堆栈), 操作数2))

-- 循环判断尾 (优先级 = “>”)

--> 数据 = 到文本 (到数值 (表达式))

表达式 = 文本替换 (表达式, 1, 取文本长度 (数据), )

数据入栈 (数据堆栈, 到数值 (数据))

负头 = 假

数据 = 选择 (数据 = “#”, “#”, 取文本左边 (表达式, 1))

-- 判断循环尾 ()

返回 (到文本 (数据出栈 (数据堆栈)))





整个子程序使用了栈的操作，并调用了其他的一些子程序。程序思路：读取编辑框中的内容并在编辑框内容的最后加入一个“#”号，用来判断算式的结束。然后从左到右对算式中的每个字符进行分析，如果字符为算数，则将数字压入数据堆栈中；如果是算符，则调用“比较优先级”子程序（该子程序用来比较算符的优先级），如果算符的优先级低，则将算符压入算符堆栈中，如果算符的优先级比“算符堆栈”的栈顶算符的优先级高，则将数据堆栈中的 2 个数弹出，和算符堆栈中栈顶算符弹出，并进行计算，将计算结果压入数据堆栈中。不断的循环进行上述的工作，当取得的符号为“#”号，即算式结束，则跳出循环并将最后的计算结果返回。

程序中入栈和出栈都使用了子程序来实现，还是用到了“比较优先级”子程序和“计算”子程序，使用到的子程序代码如下。

数据堆栈中入栈和出栈子程序如下：

子程序名	返回值类型	公开	备注		
数据出栈	整数型				
参数名	类型	参考	可空	数组	备注
堆栈	栈				

变量名	类型	静态	数组	备注
节点	节点			
数据	整数型			

堆栈.弹出(节点)

节点.取数值(“数据”，数据)

返回(数据)

子程序名	返回值类型	公开	备注		
数据入栈	逻辑型				
参数名	类型	参考	可空	数组	备注
堆栈	栈				
数据	整数型				

变量名	类型	静态	数组	备注
节点	节点			

节点.加入属性(“数据”，数据)

返回(堆栈.压入(节点))

算符堆栈中入栈和出栈子程序如下：

子程序名	返回值类型	公开	备注		
算符出栈	文本型				
参数名	类型	参考	可空	数组	备注
堆栈	栈				

变量名	类型	静态	数组	备注
节点	节点			
算符	文本型			

堆栈.弹出(节点)

节点.取文本值(“算符”，算符)

返回(算符)

子程序名	返回值类型	公开	备注		
算符入栈	逻辑型				
参数名	类型	参考	可空	数组	备注
堆栈	栈				
算符	文本型				

变量名	类型	静态	数组	备注
节点	节点			

节点.加入属性(“算符”，算符)

返回(堆栈.压入(节点))

取数据堆栈和算符堆栈栈顶元素的子程序如下：



子程序名	返回值类型	公开	备 注		
取数据栈顶	整数型				
参数名	类 型	参考	可空	数组	备 注
堆栈	栈				

变量名	类 型	静态	数组	备 注
节点	节点			
数据	整型			

堆栈, 取栈顶节点 (节点)

节点.取数值 (“数据”, 数据)

返回 (数据)

子程序名	返回值类型	公开	备 注		
取算符栈顶	文本型				
参数名	类 型	参考	可空	数组	备 注
堆栈	栈				

变量名	类 型	静态	数组	备 注
节点	节点			
算符	文本型			

堆栈, 取栈顶节点 (节点)

节点.取文本值 (“算符”, 算符)

返回 (算符)

“比较优先级”和“计算”子程序如下:

子程序名	返回值类型	公开	备 注		
比较优先级	文本型				
参数名	类 型	参考	可空	数组	备 注
算符1	文本型				
算符2	文本型				

```

判断 (算符1 = "+" 或 算符1 = "-")
  返回 (选择 (算符2 = "+" 或 算符2 = "-" 或 算符2 =
    "=") 或 算符2 = "#", ">", "<"))
  判断 (算符1 = "*" 或 算符1 = "/")
  返回 (选择 (算符2 = "(", "<", ">"))
  判断 (算符1 = "(")
  返回 (选择 (算符2 = ")") 或 算符2 = "#", "=",
    "<"))
  判断 (算符1 = ")")
  返回 (">")
  判断 (算符1 = "#")
  返回 (选择 (算符2 = "#", "=", "<"))
  返回 ("<")

```

子程序名	返回值类型	公开	备 注		
计算	整数型				
参数名	类 型	参考	可空	数组	备 注
数据1	整数型				
算符	文本型				
数据2	整数型				

```

-- 判断 (算符 = “+” )
-- 返回 (数据1 + 数据2)
-- 判断 (算符 = “-” )
-- 返回 (数据1 - 数据2)
-- 判断 (算符 = “*” )
-- 返回 (数据1 × 数据2)
-- 判断 (算符 = “/” )
-- 返回 (数据1 ÷ 数据2)

```





## 第二十一章 数据操作支持库

数据操作支持库实现了数据压缩解压、数据完整性校验、数据加解密支持。

本章主要以数据完整性校验为重点，解释和运行各命令的使用方法。在开始编写程序之前，首先了解一些相关的加密理论和加密技术。

### 21.1 加密技术

为了保证网络上传递的信息或数据的安全，通常采用加密和认证的方法。

#### 21.1.1 数据加密

常用数据加密算法有两种：对称密钥加密、非对称密钥加密。用于保证数据的保密性、完整性和真实性。

##### (1) 对称密钥加密：

早期的加密系统是基于对称加密理论（即单密钥加密理论），其特点是加密密钥和解密密钥可以互相推导，发送者和接收者在安全通信之前需要商定一个（相同的/对称的）密钥。对称加密的安全性依赖于密钥，泄露密钥意味着任何人都能对信息进行加/解密。

##### (2) 非对称密钥加密（公开密钥加密）：

常用的公开密钥算法有 RSA、DSA 等。使用公开密钥算法的用户同时拥有公钥和私钥。私钥不能通过公钥计算出来。私钥由用户自己持有，公钥以明文发送给任何人，公开密钥理论解决了对称加密系统的密钥交换问题。

为了实现数据的加密传输，公开密钥算法提供了安全的对称算法密钥交换机制（注意：这里不是对称密钥加密），数据使用对称算法加密传输。

两个用户（A 和 B）使用公开密钥理论进行密钥交换的基

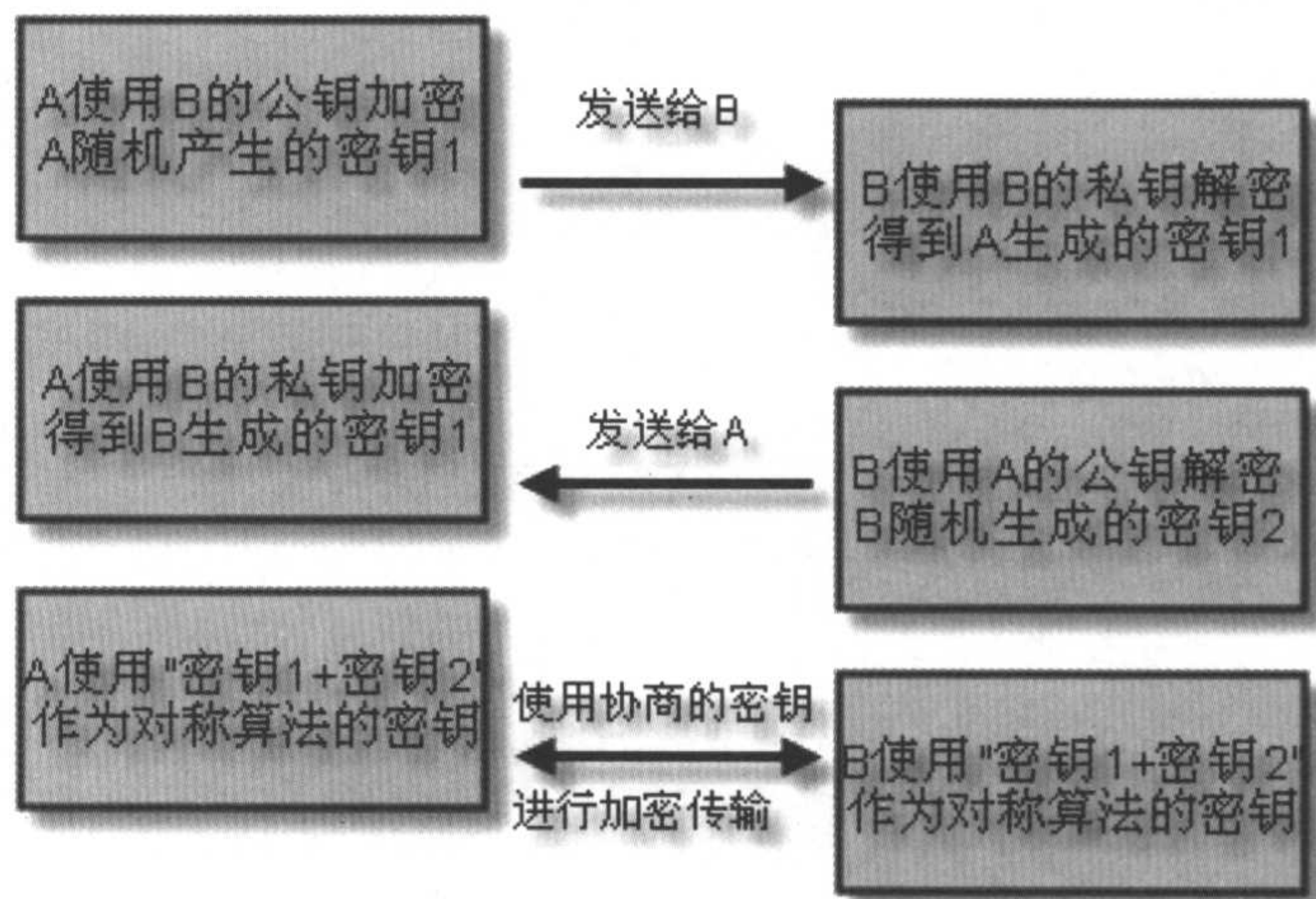


图 21-1 公开密钥理论



本过程如图 21-1 所示。

在对称算法密钥的协商过程中，密钥数据使用公钥加密。在保证私钥安全的前提下，攻击者即使截获传输的信息也不能得到加密算法的密钥，这就保证了对称算法密钥协商的安全性。

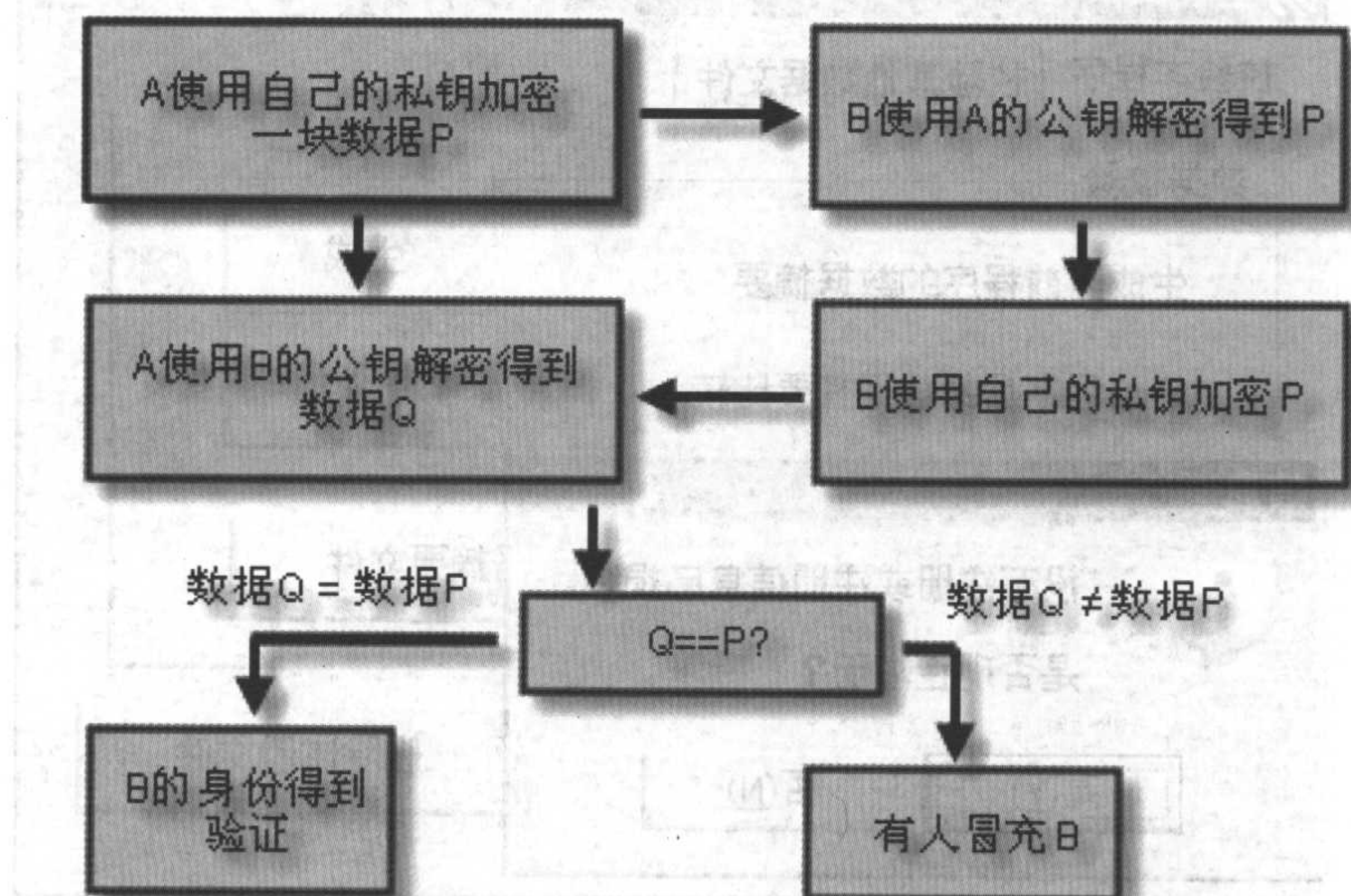
### 21.1.2 数据认证

数据认证能够实现对原始数据的鉴别和验证，保证数据的完整性、权威性和发送者对所发数据的不可抵赖、冒充、篡改等问题。其中也离不开对公开密钥加密技术的应用。

首先介绍消息摘要算法。消息摘要算法实际上就是一个单向散列函数（它是从明文到密文的不可逆函数，只能加密不能还原），公式将一个不同长度的数据转换成一个数字串（数据摘要），该函数不需要密钥，公式决定了数据摘要的长度。数据摘要和非对称加密一起，提供数字签名的方法。常用的消息摘要算法有 MD5、SHA 等。

数字签名是公开密钥加密技术的一种应用，是指用发送方的私有密钥加密数据摘要，然后将其与原始的信息附加在一起，合称为数字签名。其使用方式是：数据的发送方从数据文本中生成一个 128 位或 160 位的单向散列值（或数据摘要），并用自己的私有密钥对这个散列值进行加密，形成发送方的数字签名；然后，将这个数字签名作为数据的附件和数据一起发送给数据的接收方；数据的接收方首先从接收到的原始数据中计算出 128 位的散列值（或数据摘要），接着再用发送方的公开密钥来对数据附加的数字签名进行解密；如果这两个散列值相同，那么接收方就能确认该数字签名是发送方的。

下面是一个很简单的身份验证的方法（A 验证 B 的身份）。如图 21-2 所示。



21-2 身份验证方法

同样的原理，公开密钥算法可以进行数据的签名和验证。A 需要对一块数据签名，A 只需要使用自己的私钥加密该数据就可以完成签名。A 把数据和数据签名（私钥加密的结





果)一起发送给 B, B 使用 A 的公钥解密签名, 然后和数据进行比较, 如果相同则该签名确实是 A 签署的, 并且数据没有被篡改。

## 21.2 数据校验

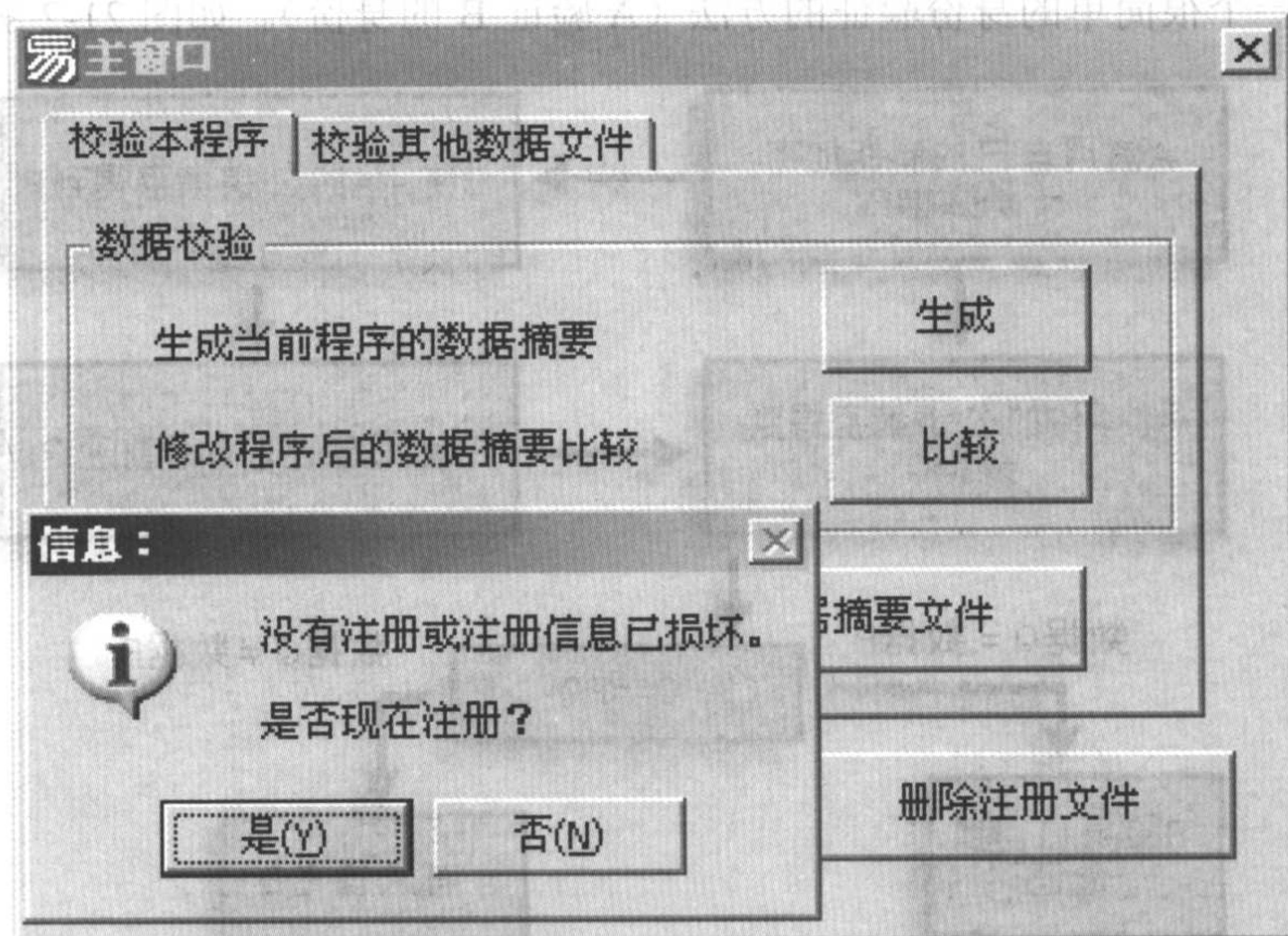
下面用一个简单的软件注册和注册码生成过程, 介绍“数字签名( )”和“签名验证( )”命令的使用方法。

数字签名用作软件注册系统的实现步骤:

1. 欲注册用户提供注册信息(如硬件代码、用户姓名等);
2. 软件作者使用自己的私钥对该注册信息进行数字签署得到签署结果文本;
3. 将此签署结果文本作为注册钥匙文件发送给用户;
4. 在用户端的软件使用相同的用户信息、注册钥匙文件及软件作者的公钥进行签名验证, 如果通过表明已经注册, 否则表示未注册。使用本方法在私钥未泄露的前提下, 可以绝对避免破解者做出软件的注册机。

打开随书光盘本章目录中的“数据校验注册.e”。

由于是首次运行, 所以随主窗口一同弹出一个信息框, 提示此程序还没有注册。如图 21-3 所示。

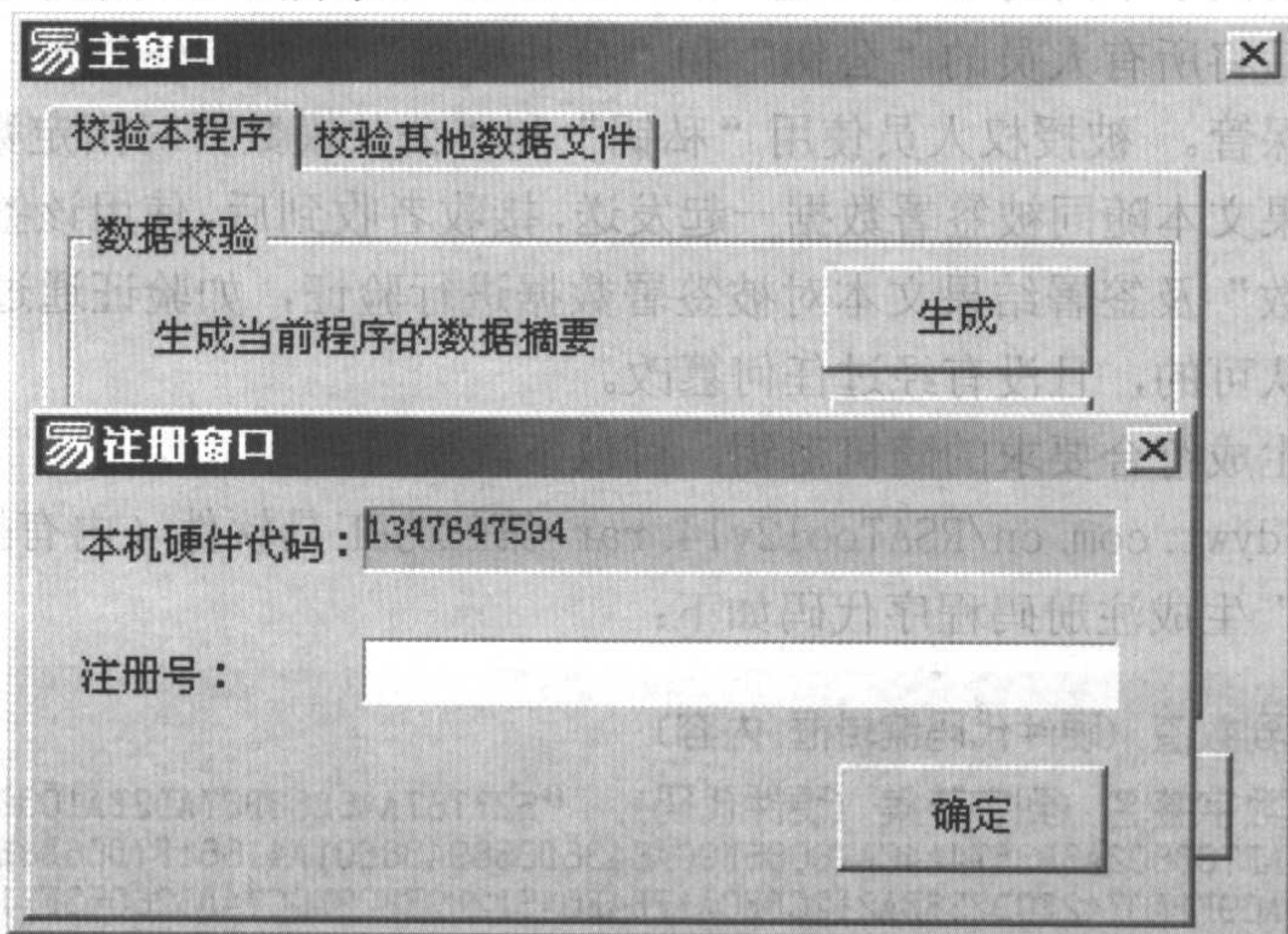


21-3 注册信息提示

按“否”钮, 程序将被关闭。在这里按“是”钮, 进入程序注册窗口, 程序同时取出

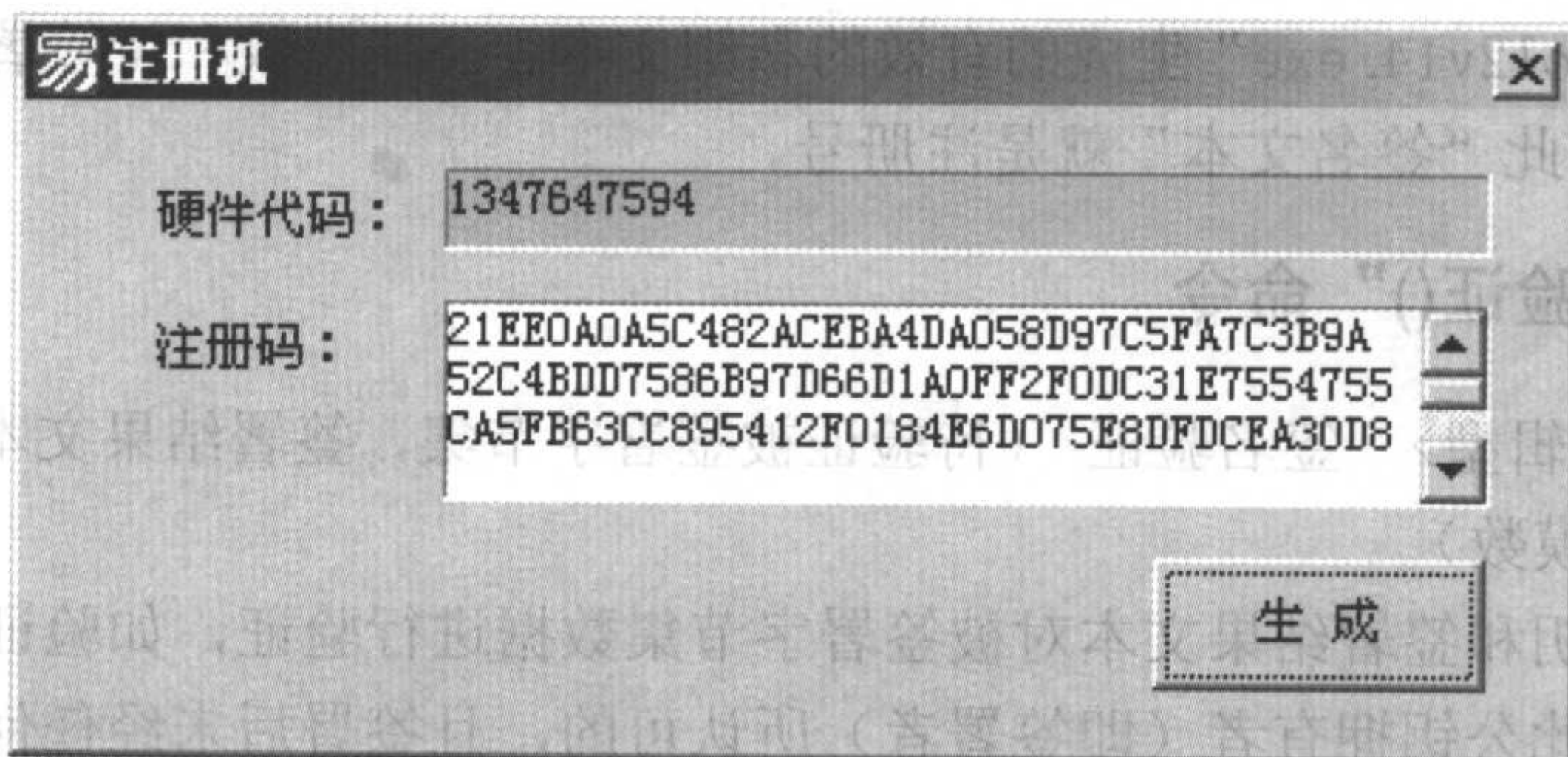


本机的硬件代码。如图 21-4 所示。



21-4 程序注册窗口

然后，打开随书光盘本章目录中的“注册机.e”。将硬件代码复制到上面的编辑框中，按“生成”按钮，在下面的编辑框中便生成需要的注册码。如图 21-5 所示。



21-5 注册机窗口

最后将注册码粘贴到前一程序的注册号框中，按“确定”按钮，注册就成功了。当程序关闭时，注册码（就是“签名文本”）被写到配置文件中，以供程序每次运行前进行注册效验。结束“注册机.e”程序。

再次运行“数据校验注册.e”，主窗口标题提示程序已经注册成功。

这个功能使用“数据完整性校验”的“数字签名（）”和“签名验证（）”命令，达到控制不同级别的用户使用程序的权限。比如使用一个硬件代码计算出的注册码，只能在硬件代码相匹配的机器上注册成功。

### 21.2.1 “数字签名（）”命令

语法：文本型 数字签名（待签署字节集数据，签署者私钥文本，签署者公共模数）

使用 RSA 非对称密钥算法对指定字节集数据进行签署，支持 32 到 4096 之间任意合法的 RSA 位数，返回签署后的结果文本。附一、电子签名系统简要实现方法：首先给所有具





有签署权利的人员授予不同的 RSA 钥匙, RSA 钥匙由“公钥”、“私钥”、“公共模数”三部分组成。然后将所有人员的“公钥”和“公共模数”成对向外公布,“私钥”由被授予人员自行妥善保管。被授权人员使用“私钥”和“公共模数”对指定数据进行签署,然后将签署后的结果文本随同被签署数据一起发送,接收者收到后,使用该签署者公开的“公钥”和“公共模数”及签署结果文本对被签署数据进行验证,如验证通过则说明该数据必定是此签署者所认可的,且没有经过任何篡改。

注意:为了生成符合要求的随机密钥,可以下载使用:

<http://www.dywt.com.cn/RSATool2v14.rar> 第三方工具软件(内有操作说明)。

“注册机.e”生成注册码程序代码如下:

硬件代码 = 到数值 (硬件代码编辑框.内容)

签名文本 = 数字签名 (到字节集 (硬件代码), “527767A46E83B37AD2BA103633B9C97638D03FAF7CB602A6D3B74A4CA78085F9CA6436D8589438E01A41861F7D06B8B7131F28CD11E29A09F1A0742E0D238BA212CBECA1FB4E04512B9EE22DOC74D626053F71F785D2F8629752D59E896549EB38A1972F06C3F858DA8DBC1F36386D221B1E1FBA1770A98A96B859502358D79F”, “B3DDC117710454E75D91051A49E9D00A2CF4A27BB1B4148A1EC82C9E0B3B590BA61FOA9CE7B9451DAC9F1B366E78AF8C98080531A5920DD29214552AB6C821A13272E4F8551E727F45CA9D8D98EB164D668399486A30A8DAA809CD39B31B4126D3B495FDCC6A174061EA6546292FA3B75752081E3FF080B358AE0428DCDF5B61”)

使用“RSATool2v14.exe”生成的有效的私钥文本、公共模数和待签署的数据生成唯一的“签名文本”,此“签名文本”就是注册号。

### 21.2.2 “签名验证()”命令

语法: <逻辑型> 签名验证 (待验证被签署字节集, 签署结果文本, 签署者公钥文本, 签署者公共模数)

使用指定公钥和签署结果文本对被签署字节集数据进行验证,如验证通过则说明该字节集数据必定是此公钥拥有者(即签署者)所认可的,且签署后未经任何篡改。验证通过返回真,否则返回假。

“数据校验注册.e”程序中验证代码如下:

```
如果 (签名验证 (到字节集 (取硬盘特征字 ()), 配置项文本, “2F6DB”, “B3DDC117710454E75D91051A49E9D00A2CF4A27BB1B4148A1EC82C9E0B3B590BA61FOA9CE7B9451DAC9F1B366E78AF8C98080531A5920DD29214552AB6C821A13272E4F8551E727F45CA9D8D98EB164D668399486A30A8DAA809CD39B31B4126D3B495FDCC6A174061EA6546292FA3B75752081E3FF080B358AE0428DCDF5B61”) ≠ 真)
```

使用和私钥文本成对的公钥文本、相同的公共模数以及唯一的“签名文本”就可以验证待验证数据的有效性。这只是一个简单的注册方法,大家也可以使用其他的命令和方法在生成和效验时对需要保密的数据进行更为复杂的处理。比如本支持库提供的数据压缩和解压、数据加解密,但要注意的是各命令的使用顺序,当一段数据先被压缩,后被加密之后,必须先解密后解压才能还原,否则生成的数据是错误的。

下面介绍“取数据摘要()”命令的应用方法。



21.2.3 “取数据摘要()” 命令

语法：文本型 取数据摘要（字节集数据）

返回一段字节集数据的 MD5 数据摘要编码文本。不同数据的 MD5 码都不一样，因此本命令可以用作保证重要数据不会被篡改。

仍然使用随书光盘本章目录中的“数据校验注册.e”。

窗口运行界面如图 21-6 所示。

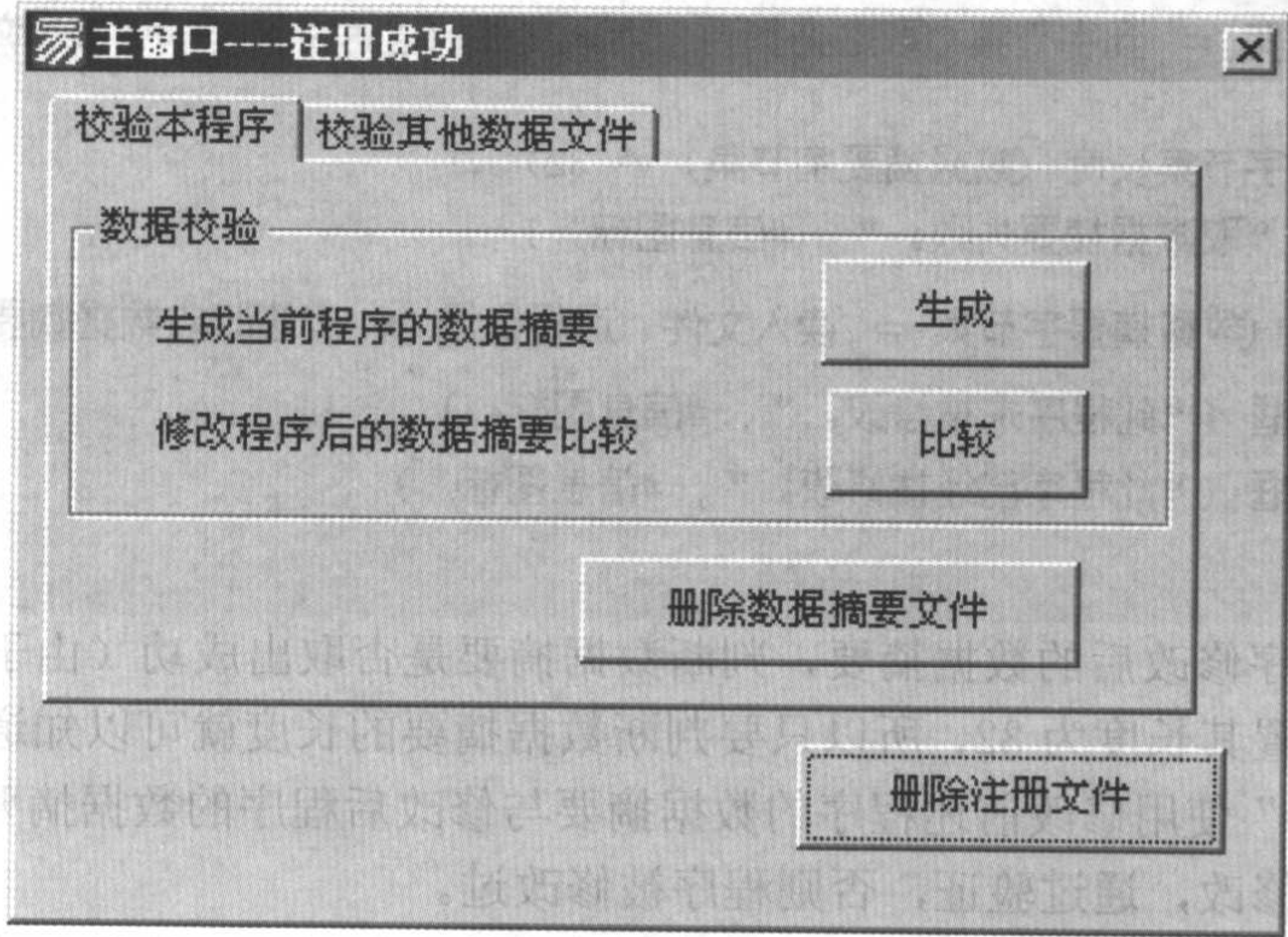


图 21-6 程序窗口界面

“效验本程序”子夹的功能用来校验本程序是否被修改。使用方法如下：

- 1. 运行程序，点击“生成”按钮，生成本程序的数据摘要，并写到一个文件中。程序代码如下：

子程序名	返回值类型	公开	备注
_生成按钮_被单击			

变量名	类型	静态	数组	备注
数据摘要字节集	字节集			

```
数据摘要字节集 = 到字节集 (取数据摘要 (读入文件 (运行目录 + “数据效验注册.e”)))
-- 如果 (取字节集长度 (数据摘要字节集) ≠ 32)
-- 信息框 (“取数据摘要失败。”, #信息图标, )
-- 如果真 (写到文件 (运行目录 + “修改前数据摘要.id”, 取字节集左边 (数据摘要字节集, 32)) = 真)
-- 信息框 (“成功写出数据摘要。”, 0, )
-- 返回 0
-- 信息框 (“写出数据摘要失败。”, 0, )
```

- 2. 关闭被运行的程序，在程序设计窗体上添加任意的一个组件（目的是改变本程序的数据结构）后保存(Ctrl + S 键)。





3. 按“F5”键再次运行程序，按“比较”按钮，弹出信息框提示本程序已经被修改。程序代码如下：

子程序名	返回值类型	公开	备注
_比较按钮_被单击			

变量名	类型	静态	数组	备注
数据摘要字节集	字节集			

数据摘要字节集 = 到字节集 (取数据摘要 (读入文件 (运行目录 + “数据效验注册.e”)))

如果 (取字节集长度 (数据摘要字节集) ≠ 32)

信息框 (“取数据摘要失败。”, #信息图标, )

如果 (数据摘要字节集 = 读入文件 (运行目录 + “修改前数据摘要.id”))

信息框 (“此程序未被修改。”, #信息图标, )

信息框 (“此程序已经被修改。”, #信息图标, )

首先取出程序修改后的数据摘要，判断数据摘要是否取出成功（由于在生成符合条件的数字签名时设置其长度为 32，所以只要判断数据摘要的长度就可以知道是否取出成功）。后一个“如果（）”使用修改前述程序的数据摘要与修改后程序的数据摘要进行比较，相等说明程序没有被修改，通过验证，否则程序被修改过。

“效验其他数据文件”子夹显示其他数据效验的功能界面。使用方法如下：

点击“生成数据摘要”按钮，选择文件，取出文件的数据摘要，程序自动将数据摘要保存到文件的同一目录中，后缀名为 id 的文件。然后，点击“选择数据摘要”按钮，选择 id 文件，程序可以使用此文件中的数据与相对应文件的数据摘要进行比较。程序代码参见例程。

以上只是一个简单例程，在实际应用中，数据加密、数据效验、数据安全的操作和实现会更为复杂。

主 窗 体	数据摘要	数据摘要	数据摘要	数据摘要
主 窗 体	数据摘要	数据摘要	数据摘要	数据摘要



## 第二十二章 数值计算支持库

从易语言 3.8 版后，推出了数值运算支持库，用来进行各种高级的数学运算，它可以进行的计算有：复数运算、矩阵运算、傅立叶变换、概要统计、微积分、联立方程、曲线拟和、大数运算等等。

### 22.1 数值计算支持库简介

数值运算支持库包括的数据类型有：复数运算、矩阵运算、傅立叶变换、概要统计、微积分、联立方程、曲线拟和、大数、其他计算、算式解析。

本支持库依然采用了调用对象方法的形式，在使用支持库中数据类型的方法，即对象的方法，首先要新建一个该数据类型的变量，然后再使用“对象.方法()”的格式来调用科学运算的各种命令，例如要调用“复数运算”数据类型中的复数相加方法：

变量名	类型	静态	数组	备注
复数1	复数运算			
复数2	复数运算			
结果	复数运算			

复数1.置实数 (到数值 (复1实数编辑框.内容))

复数1.置虚数 (到数值 (复1虚数编辑框.内容))

复数2.置实数 (到数值 (复2实数编辑框.内容))

复数2.置虚数 (到数值 (复2虚数编辑框.内容))

结果 = 复数1.复数相加 (复数1, 复数2)

### 22.2 数值计算支持库的各数据类型

#### 22.2.1 复数运算

复数的概念：

形如  $a+bi$  ( $a$ 、 $b$  都是实数) 的数叫做复数。 $a$  叫做复数的实部，它的单位是 1， $bi$  叫





做复数的虚部，它的单位是  $i$ ； $b$  叫做复数的虚部系数。

复数运算数据类型中提供的命令，就是遵照复数各种运算的规则进行复数运算。其中包括的命令有：复数相加、复数相减、复数相乘、复数相除、求复数指数、求复数长度、求复数极角、置实数、置虚数、取实数、取虚数。

这里注意，复数运算中的参数，都要提供一个“复数运算”数据类型的数据，返回值也是一个“复数运算”数据类型的数据。在进行复数运算前，先要使用置实数和置虚数命令，设置要进行运算的复数的实部和虚部，然后使用复数运算的命令进行计算，例如进行算式： $(1000+200i) \times (1000+100i)$  的计算，使用代码：

变量名	类型	静态	数组	备注
复数1	复数运算			
复数2	复数运算			
运算结果	复数运算			

复数1.置实数 (1000)

复数1.置虚数 (200)

复数2.置实数 (1000)

复数2.置虚数 (100)

运算结果 = 复数1.复数相乘 (复数1, 复数2)

运算结果的实部和虚部用“取实数”和“取虚数”方法取得。

## 22.2.2 矩阵运算

定义： $m \times n$  个数排列成  $m$  行（横向）、 $n$  列（纵向）的矩形数表：

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

称为  $m \times n$  矩阵，简记为  $A = (a_{ij})_{m \times n}$ ，其中  $a_{ij}$  为  $A$  的第  $i$  行第  $j$  列的元。

易语言矩形运算数据类型提供的方法有：矩阵相乘、矩阵相加、实矩阵逆、矩阵转置、求矩阵特征、矩阵标量乘积。

例如求 2 个 2 行 2 列矩阵之和，2 个矩阵为：

$$\begin{bmatrix} 100 & 200 \\ 202 & 323 \end{bmatrix} + \begin{bmatrix} 201 & 223 \\ 245 & 454 \end{bmatrix}$$



矩阵运算程序实现代码为:

子程序名	返回值类型	公开	备注
矩阵运算			

变量名	类型	静态	数组	备注
矩阵运算	矩阵运算			
矩阵1	双精度小数型		2,2	
矩阵2	双精度小数型		2,2	
计算结果	双精度小数型		0	

矩阵1 [1] [1] = 100  
矩阵1 [1] [2] = 200  
矩阵1 [2] [1] = 202  
矩阵1 [2] [2] = 323  
矩阵2 [1] [1] = 201  
矩阵2 [1] [2] = 223  
矩阵2 [2] [1] = 245  
矩阵2 [2] [2] = 454  
计算结果 = 矩阵运算. 矩阵相加 (矩阵1, 矩阵2)

相加运算后，所得的矩阵将以一个多维数组的形式返回，返回的多维数组用矩阵的形式表现为：

301

447

,

423

777

22.2.3 傅立叶变换

在 1807 年，法国数学家傅立叶（Jean Baptiste Joseph Fourier）声称，任何可重复的波形（或周期性的函数）就如同音叉的声波一样都能以一个不同频率的正弦波和余弦波的无限和的形式来表达。（余弦波就是正弦波向前位移四分之一周期。）傅立叶转换现在已经应用于各个领域，如物理和工程学、数字信号处理等等。

在了解傅立叶变换前，可以先了解一下单位根的性质。

方程： $x^n - 1 = 0$

有 n 个根称为 n 次单位根。如果 w 是 n 次单位根，那么， $w^k$  (k 为整数) 也一定是 n 次单位根，这是因为

$$(w^k)^n = w^{nk} = (w^n)^k = 1^k = 1$$

如果 w 是 n 次单位根，且

$$1, w, w^2, \Lambda, w^{n-1}$$





都不相同,则称  $w$  为  $n$  次单位元根。显然,方程  $x^n - 1 = 0$  的一个  $n$  次单位元根为

$$w = e^{j2\pi/n}$$

并且

$$w = e^{-j2\pi/n}$$

也是一个  $n$  次单位根。这就是说,在离散傅里叶变换中定义的  $w$  正好是一个  $n$  次单位元根。

由上看出,  $n$  次单位元根不是惟一的,但也并不是所有的  $n$  次单位根都可以称为  $n$  次单位元根。例如, 1 是  $n$  次单位根,但不是  $n$  次单位元根。

下面简单介绍一下离散傅里叶变换的一个公式。设函数  $f(x)$  在区间  $[a, b]$  上取  $n$  个等分采样点

$$x_k = k\Delta x, k = 0, 1, \Lambda, n-1$$

其函数值为:

$$f(x_k) = f(k\Delta x), k = 0, 1, \Lambda, n-1$$

经过换算得出

$$C_m = \frac{1}{n} \sum_{k=0}^{n-1} f(kT/n) e^{-j(2\pi k/n)m}, m = 0, 1, \Lambda, n-1$$

可以称上面公式为周期函数  $f(t)$  在  $n$  个等分点上的离散傅里叶变换。其中  $f(kT/n)$  为第  $k$  个分点上的采样值。通常  $C_m$  是个复数。

关于更多的傅里叶相关内容,可以查看相关书籍。易语言中提供了 5 个方法进行傅里叶变换的运算,包括:求傅立叶变换、求傅立叶反变换、窗口傅立叶、二维傅立叶变换、求能谱周期图。调用方法和上面介绍的其他数据类型的方法相同。

## 22.2.4 微积分

微积分是高等数学中的重要内容之一,应用的领域也很广。介绍微积分的书籍比较多,这里不做具体介绍。

易语言中提供的微积分数据类型有 3 个方法组成,包括:离散化数值积分、函数数值积分、一阶微分方程。例如:

变量名	类型	静态	数组	备注
微积分	微积分			

信息框 (微积分.函数数值积分 (100, 1, 100), 0, )

## 22.2.5 概要统计

概要统计用来对一个数据集中的数据进行一系列的数据指数的统计。



易语言中概要统计数据类型提供了 1 个方法：统计数据概要，该方法对数据集的每一列计算概要统计，包括最大值、最小值、范围、总和、平均值、方差、标准偏差、众数和平均值的标准误差。例如对 1-100 的整数集合进行概要统计：

变量名	类型	静态	数组	备注
统计数据	概要统计			
原数据	双精度小数型		10, 10	
最小值	双精度小数型		10	
最大值	双精度小数型		10	
范围值	双精度小数型		10	
总和	双精度小数型		10	
平均值	双精度小数型		10	
方差值	双精度小数型		10	
标准偏差值	双精度小数型		10	
标准误差	双精度小数型		10	
众数值	双精度小数型		10	
循环变量	整数型			

---▶ 计次循环首 (100, 循环变量)

原数据 [循环变量] = 循环变量

--- 计次循环尾 0

统计数据.统计数据概要 (原数据, 最小值, 最大值, 范围值, 总和, 平均值, 方差值, 标准偏差值, 标准误差, 众数值)

统计后的各结果指数，存放在相应的变量中，可以读取变量的值来取得计算的结果。

## 22.2.6 联立方程

联立方程数据类型，提供了 1 个计算方法：线性方程组。线性方程组是《线性代数》中的一部分，可以查看相关书籍来了解线性方程组的定义及相关内容，这里只介绍易语言中“线性方程组”方法的简单使用。例如，系数矩阵：

$$\begin{bmatrix} 1 & 2 \\ 4 & 2 \end{bmatrix}$$

对其进行线性方程求解：





变量名	类型	静态	数组	备注
联立方程	联立方程			
x变量的系数	双精度小数型		2, 2	
y向量	双精度小数型		2	
x变量的解	双精度小数型		2	
x变量系数逆阵	双精度小数型		2, 2	
x矩阵行列式	双精度小数型			

x变量的系数 [1] = 1

x变量的系数 [2] = 2

x变量的系数 [3] = 4

x变量的系数 [4] = 2

联立方程. 线性方程组 (x变量的系数, y向量, x变量的解, x变量系数逆阵, x矩阵行列式)

### 22.2.7 曲线拟和

曲线拟和是用连续曲线近似地刻画或比拟平面上离散点组所表示的坐标之间的函数关系的一种数据处理方法。用解析表达式逼近离散数据的一种方法。在科学实验或社会活动中，通过实验或观测得到量  $x$  与  $y$  的一组数据对  $(x_i, y_i)$  ( $i=1, 2, \dots, m$ )，其中各  $x_i$  是彼此不同的。人们希望用一类与数据的背景材料规律相适应的解析表达式， $y=f(x, c)$  来反映量  $x$  与  $y$  之间的依赖关系，即在一定意义下“最佳”地逼近或拟合已知数据。 $f(x, c)$  常称作拟合模型，式中  $c=(c_1, c_2, \dots, c_n)$  是一些待定参数。当  $c$  在  $f$  中线性出现时，称为线性模型，否则称为非线性模型。有许多衡量拟合优度的标准，最常用的一种做法是选择参数  $c$  使得拟合模型与实际观测值在各点的残差(或离差) $e_k=y_k-f(x_k, c)$  的加权平方和达到最小，此时所求曲线称作在加权最小二乘意义下对数据的拟合曲线。有许多求解拟合曲线的成功方法，对于线性模型一般通过建立和求解方程组来确定参数，从而求得拟合曲线。至于非线性模型，则要借助求解非线性方程组或用最优化方法求得所需参数才能得到拟合曲线，有时称之为非线性最小二乘拟合。

易语言中曲线拟和数据类型提供了 3 种运算方法，包括：二乘曲线拟合、三次多项式方程、三次样条插值。

### 22.2.8 大数

大数运算用来对特大数进行计算，大数的位数没有限制，仅限于内存及硬盘空间大小，例如对超过万亿以上的数值进行计算。易语言中大数支持库中提供了 42 个方法用来对大数进行计算，大家可以直查看支持库列表。方法中包含了对大数的运算、比较、导入和导出等多方面的命令。下面介绍几个常用方法：

由于大数运算需要很长的一段数字，所以本支持库采取导入数字文本的方式，来导入运算数据，也可以导入数字文本文件。可以使用的方法有：“导入文本文件 ()”、“导入文本 ()”；同样，运算后的数据可以使用“导出文本 ()”方法导出数字文本。

例如：



变量名	类型	静态	数组	备注
大数1	大数			
大数2	大数			
结果	大数			

结果 = 大数1.加 (大数2)

## 22.2.9 其他计算

其他计算数据类型中，提供了 23 种方程和公式的计算，包括：“求正切、求双曲余弦、求双曲正弦、求双曲正割、反正切、求伽玛、求伽玛对数、求不完全伽玛、求不完全伽玛 2、求贝它、求不完全贝它、求贝塞尔、求修正贝塞尔、求修正贝塞尔 R、求高斯误差、求误差余、求误差实部、求误差虚部、求厄密多项式、求勒让德方程系数、求拉格尔、求雅各比方程、求车切多项式”。可以根据需要，选择欲进行的计算方法，该数据类型的使用方法和前面介绍的相同。

## 22.2.10 算式解析

算式解析数据类型，提供了 1 个方法：求积分表达式。本方法可求各种复杂的积分表达式，如  $3 \cdot \exp(x) \cdot \tan(x) / (8 + \log(x))$ ，并且可以检查出表达式的错误，如果积分表达式有误或计算失败，本方法将返回假，计算成功本方法返回真。

# 22.3 大数计算器

下面来编写一个简单的大数计算器。

首先新建一个易程序，然后进行界面的设计，如图 22-1 所示：

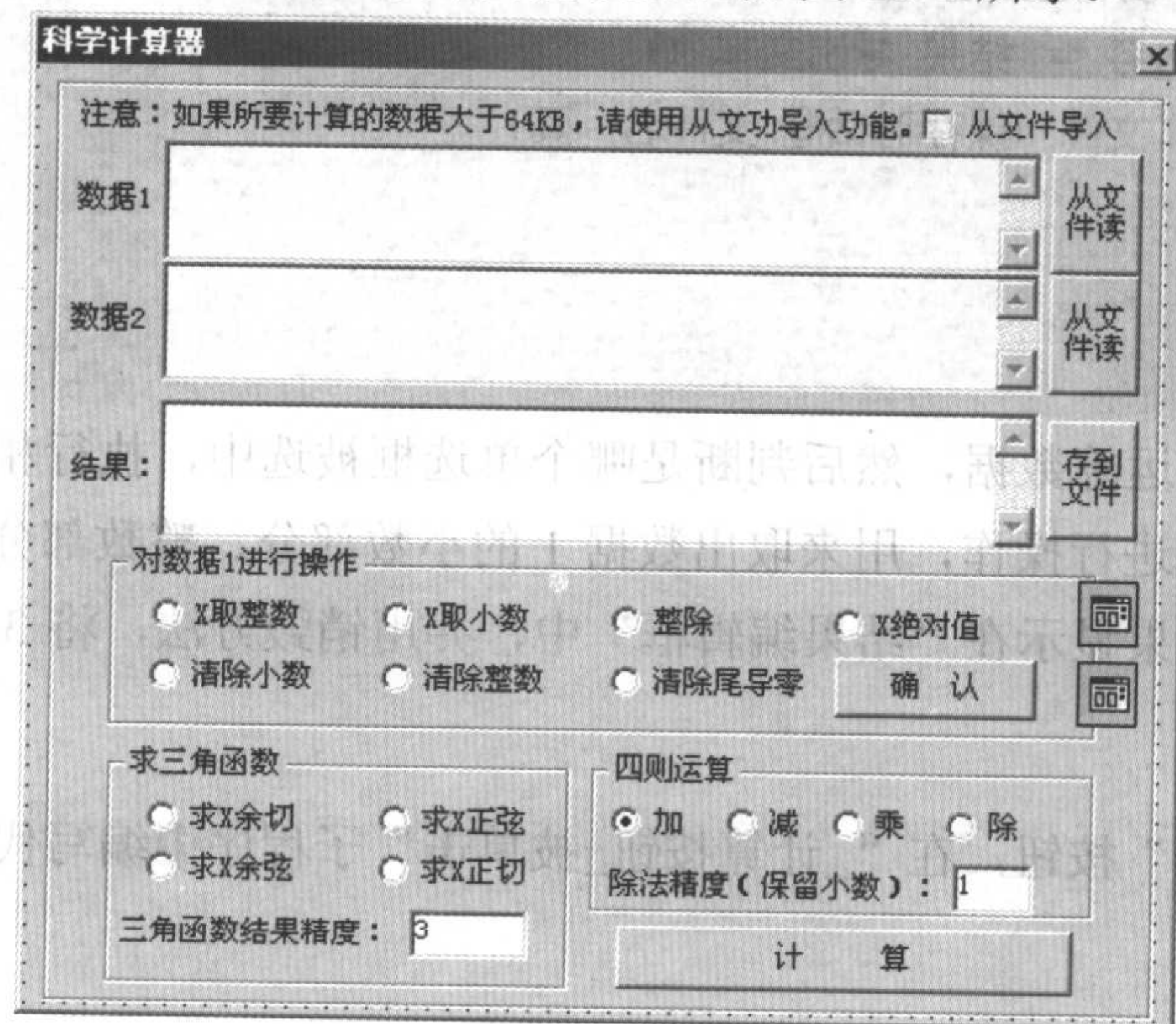


图 22-1 大数计算器界面





程序中，使用单选框来选择要进行的操作和计算。定义 3 个数据类型为“大数”的变量，名称为“数据 1”、“数据 2”、“结果”。双击确认按钮，在“\_确认按钮\_被单击”子程序中编写代码：

子程序名	返回值类型	公开	备注
_确认按钮_被单击			

```
--- 如果真 (从文件导入选择框.选中 = 假)
    数据1.导入文本 (大数1编辑框.内容)
    ▶ 判断 (取整数单选框.选中 = 真)
        结果 = 数据1.取整数 ()
    ▶ 判断 (取小数单选框.选中 = 真)
        结果 = 数据1.取小数 ()
    ▶ 判断 (绝对值单选框.选中 = 真)
        结果 = 数据1.取绝对值 ()
    ▶ 判断 (清除小数单选框.选中 = 真)
        数据1.复制到 (结果)
        结果.清除小数 ()
    ▶ 判断 (清除整数单选框.选中 = 真)
        数据1.复制到 (结果)
        结果.清除整数 ()
    ▶ 判断 (清除尾导零单选框.选中 = 真)
        数据1.复制到 (结果)
        结果.清除尾导零 ()
    ▶
    ▶ 如果 (从文件导入选择框.选中 = 假)
        结果编辑框.内容 = 结果.导出文本 ()
    ▶ 结果.导出文本文件 (保存对话框.文件名, 真)
    结果.销毁 ()
    数据1.销毁 ()
    数据2.销毁 ()
```

程序中，先导入运算数据，然后判断是哪个单选框被选中，执行相关的命令。该子程序中，都是对数据 1 进行操作，用来取出数据 1 的小数部分、整数部分或取绝对值等等。最后，将运行后的结果显示在“结果编辑框”中，并用销毁方法，将 3 个大数变量中的数据释放。

然后点击“计算”按钮，在“\_计算按钮\_被单击”子程序中编写代码：



子程序名	返回值类型	公开	备注
_计算按钮_被单击			

```

--- 如果真 (从文件导入选择框.选中 = 假)
    数据1.导入文本 (大数1编辑框.内容)
    数据2.导入文本 (大数2编辑框.内容)
    ▶ 判断 (加单选框.选中 = 真)
        结果 = 数据1.加 (数据2)
    ▶ 判断 (减单选框.选中 = 真)
        结果 = 数据1.减 (数据2)
    ▶ 判断 (乘单选框.选中 = 真)
        结果 = 数据1.乘 (数据2)
    ▶ 判断 (除单选框.选中 = 真)
        结果 = 数据1.除 (数据2, 到数值 (位数编辑框.内容))
    ▶ 判断 (整除单选框.选中 = 真)
        结果 = 数据1.整除 (数据2)
    ▶ 判断 (求正弦单选框.选中 = 真)
        结果 = 数据1.求正弦 (到数值 (三角精度编辑框.内容))
    ▶ 判断 (求余弦单选框.选中 = 真)
        结果 = 数据1.求余弦 (到数值 (三角精度编辑框.内容))
    ▶ 判断 (求正切单选框.选中 = 真)
        结果 = 数据1.求正切 (到数值 (三角精度编辑框.内容))
    ▶ 判断 (求余切单选框.选中 = 真)
        结果 = 数据1.求余切 (到数值 (三角精度编辑框.内容))
    ▶
    如果 (从文件导入选择框.选中 = 假)
        结果编辑框.内容 = 结果.导出文本 ()
        结果.导出文本文件 (保存对话框.文件名, 真)
    结果.销毁 ()
    数据1.销毁 ()
    数据2.销毁 ()

```

子程序中，首先将要进行计算的 2 个大数导入数据 1 和数据 2 中，然后判断哪个选择框被选中，然后进行相应的计算。最后将计算结果显示在“结果编辑框中”，再将 3 个大数销毁掉。

下面看一下其他子程序中的代码：

子程序名	返回值类型	公开	备注
_从文件读按钮1_被单击			

```

--- 如果真 (打开对话框.打开 () = 真)
    大数1编辑框.内容 = 打开对话框.文件名
    数据1.导入文本文件 (打开对话框.文件名, 真)

```





子程序名	返回值类型	公开	备注
_从文件读按钮2_被单击			

--- 如果真 (打开对话框. 打开 () = 真)  
大数2编辑框. 内容 = 打开对话框. 文件名  
数据2. 导入文本文件 (打开对话框. 文件名, 真)

当“从文件读按钮”被单击后，弹出通用对话框，使用“导入文本文件”方法，将选中的文本文件导入到“数据1”和“数据2”中。要注意的是，导入的文本文件必须仅为存储数字内容，否则本方法将失败。

子程序名	返回值类型	公开	备注
_保存到文件按钮_被单击			

--- 如果真 (保存对话框. 打开 () = 真)  
结果编辑框. 内容 = 保存对话框. 文件名

当“保存到文件按钮”被单击后，将保存对话框的文件名显示在结果编辑框中。

在前面讲到的“确认按钮被单击”和“计算按钮被单击”子程序中，当运算结束后，如果“从文件导入选择框”被选中，则将运算结果导出到保存编辑框显示的文件中。

例程参见随书光盘中的例程：“大数计算器. e”。

子程序名	返回值类型	公开	备注
_保存到文件按钮_被单击			



## 第二十三章 文本语音转换支持库

### 23.1 文本语音转换简介

多媒体电脑出现的以后，如果需要让电脑播放朗读文章的声音，可以把人朗读文章的声音录制成声音文件，由电脑播放。后来出现了可以朗读任意文本内容的软件，其原理是把每个单词的人声发音单独存入电脑，再根据文章内容逐个调用播放。由于声音库文件占用存储空间巨大，普通个人电脑无法胜任，因此并未普及。经过长时间的研究后，现已开发出只需要少量指令代码即可让电脑硬件模拟出真人发音，在个人电脑上朗读任意文本成为了现实。

自计算机问世以来，人们一直在努力简化指令的输入方式、提高输入效率。拨动开关、打孔机、键盘、鼠标、遥控器……电脑指令的输入方式已经发生了巨大的改变。过去限于计算机的处理速度，无法即时处理声音信息，随着计算机处理速度的大幅提高，个人电脑已经具备即时处理语音信息的能力。大家可以通过简单的语音指令控制电脑的操作，也可以口述文章由电脑自动识别录入，彻底改变了过去机械的输入方式，极大地方便了电脑操作员们的工作。

**注意：**在运行本章例程之前，必须安装微软的语音引擎组件包。该组件包官方下载网址是“<http://www.microsoft.com/speech/download/sdk51/>”，注意 SpeechSDK51 和 SpeechSDK51LangPack 两个文件都要下载安装。易语言网站下载网址是“<http://www.dywt.com.cn/edown/>”。

### 23.2 机读文本

易语言提供的“机读文本”数据类型可以将文本转换为语音由电脑输出。它将复杂的 API 调用全部封装起来，只需要几个简单的命令即可实现机读文本功能。大家先来看看相关的命令。

#### 1. “创建 ()”方法

启动系统安装的发音引擎。

#### 2. “释放 ()”方法

关闭已经启动的发音引擎。此命令与“创建”命令配合使用。





## 3. “文本到语音 ()” 方法

通过已经启动的发音引擎，把文本或者文件转换成为语音输出，成功返回“真”，失败返回“假”。

## 4. “输出声音文件 ()” 方法

通过已经启动的发音引擎，把文本或者文件转换成为语音输出到指定 WAV 文件，成功返回“真”，失败返回“假”。

## 5. “停止发音 ()” 方法

停止当前电脑发音，成功返回“真”，失败返回“假”。当前发音被停止后只能从起始重新开始识别。

## 6. “暂停发音 ()” 方法

暂停当前发音，成功返回“真”，失败返回“假”。可用“恢复发音”命令继续。

## 7. “恢复发音 ()” 方法

恢复被暂停的发音，成功返回“真”，失败返回“假”。

## 8. “设置声音大小 ()” 方法

根据参数值设置声音大小，成功返回“真”，失败返回“假”。该参数值的范围是 0 到 100 之间。

## 9. “设置语速 ()” 方法

根据参数值设置语速快慢，成功返回“真”，失败返回“假”。该参数值的范围是从 -10 到 10 之间。

## 10. “列举语音库 ()” 方法

以文本型数组方式返回当前系统所安装的所有语音库的名称。

## 11. “设置语音库 ()” 方法

指定用来发音的语音库，成功返回“真”，失败返回“假”。设置前必须停止当前发音。

打开随书光盘本章目录中的例程“文本朗读.e”，设计界面如图 23-1 所示。

窗口中的编辑框为识别文本显示区，将需要识别的文本填入其中。识别区右边为语速控制滑动条和音量控制滑动条。工具条中各按钮功能分别为：清空识别区、打开文本文件、输出至 WAV 文件、停止发音、开始/继续发音、暂停发音，组合框用来指定发音语音库。还加入了一个通用对话框用来打开和保存文件。

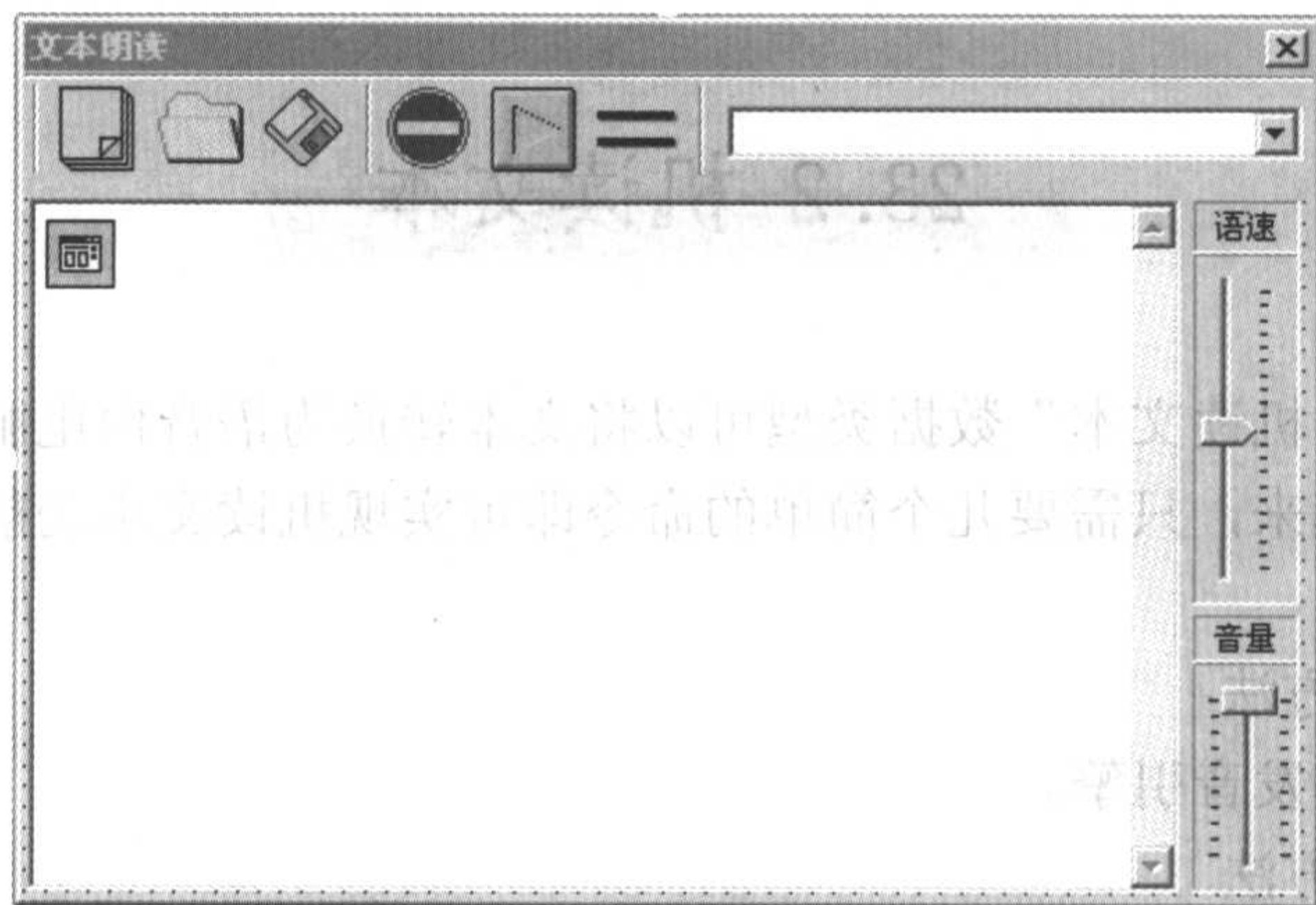


图 23-1 朗读文本界面设计窗口



下面看看相关代码。

首先定义两个程序集变量：机读，数据类型为机读文本；暂停，数据类型为逻辑型，用来表示当前处于暂停发音状态还是停止发音状态。

窗口程序集名	备注		
窗口程序集1			
变量名	类型	数组	备注
机读	机读文本		
暂停	逻辑型		

程序运行后，首先调用“\_\_启动窗口\_创建完毕”子程序。在“\_\_启动窗口\_创建完毕”子程序中，首先通过“创建”命令启动系统安装的发音引擎，然后调用“列举语音库（）”方法返回当前系统所安装的所有语音库的名称，如果没有发现安装好的语音库，则提示后退出程序。通过循环方法将所收集到的语音库名称全部加入组合框中备选，并设置第一个语音库为当前使用的语音库。语速滑块条的最小位置为 0，代表语速为-10；最大位置为 20，代表语速为 10，现在将滑块位置设为 10，代表语速为 0。

代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
文本数组	文本型		0	
容器	整数型			

机读.创建 ()

文本数组 = 机读.列举语音库 ()

容器 = 取数组成员数 (文本数组)

--- 如果真 (容器 ≤ 0)

    信息框 (“尚未安装语音引擎，请先安装。” + #换行符 + “程序退出”， 0, )

    返回 ()

--- 计次循环首 (取数组成员数 (文本数组), 容器)

    组合框1.加入项目 (文本数组 [容器], )

--- 计次循环尾 ()

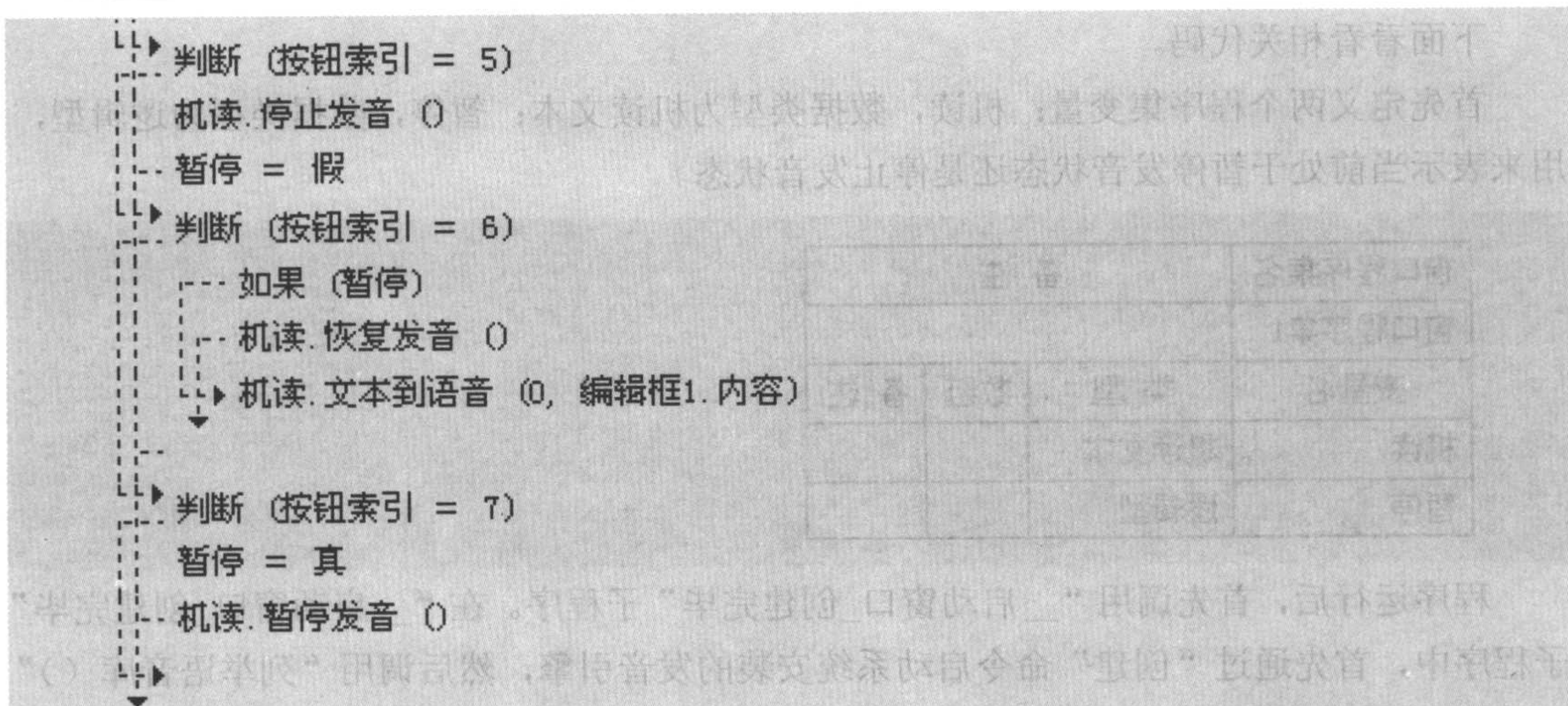
组合框1.现行选中项 = 0

机读.设置语音库 (组合框1.取项目文本 (0))

滑块条语速.位置 = 10

下面的代码是工具条中停止发音、开始/继续发音、暂停发音三个按钮的事件处理代码。通过判断逻辑型变量“暂停”的值，就能区分当前是处于停止状态还是暂停状态。





以下是指定当前语音库的代码。大家可以发现，在指定新语音库前必须先停止当前发音，否则因语音引擎被占用而无法有效指定新语音库。

子程序名	返回值类型	公开	备注
组合框1_列表项被选择			

机读. 停止发音 ()

机读. 设置语音库 (组合框1. 取项目文本 (组合框1. 现行选中项))

其余的源代码请读者自行查看理解。

需要说明的是，微软语音库有中英文之区分，中文语音库无法正确识别单词，只能单个字母识别，英文语音库无法识别中文汉字，因此在指定语音库时，一定要正确选择。安装金山词霸等带机读英文语音功能的程序，安装的都是英文语音库，如果程序不能正确发音，请检查是否正确安装了中文语音库。如图 23-2 所示。

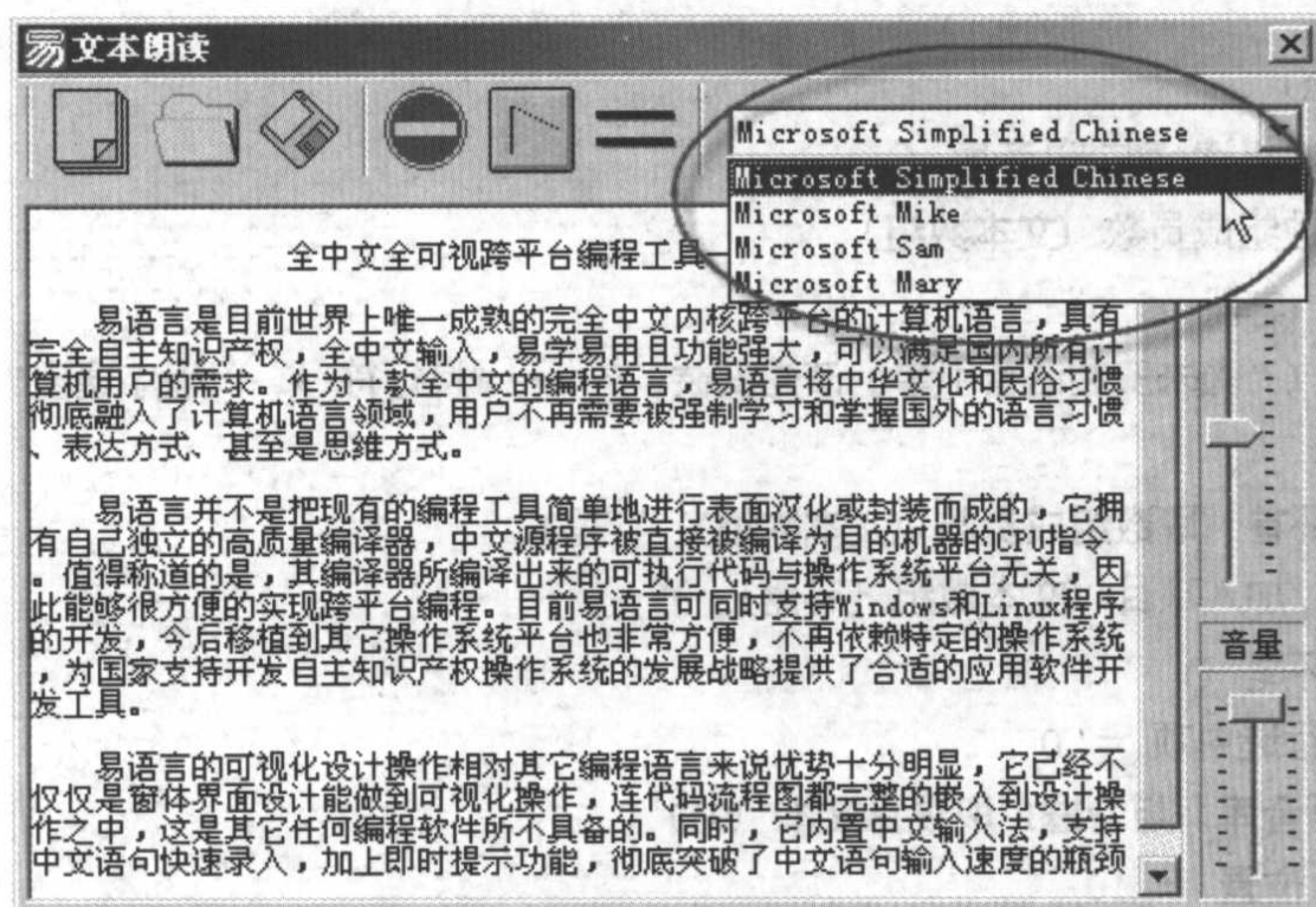


图 23-2 中文识别必需指定中文语音库



## 23.3 语音识别

易语言提供的“语音识别”组件可实现语音控制、语音录入和即时翻译等功能。每个人的音域、音调和发音习惯都不相同，为了让电脑能够更准确的识别发音者的话语，需要先对电脑进行“训练”，让电脑熟悉使用者的发音特性，了解使用者经常用到的词汇。经过一段时间的磨合后，语音识别的效果还是不错的。

下面了解一下语音识别的相关命令。

### 1. “创建（）”方法

使用指定的语音识别引擎创建语音识别区域，该区域可指定为系统范围后台识别，或仅当本程序具有系统焦点时方能识别。成功返回真，失败返回假。

### 2. “释放（）”方法

释放已经创建的语音识别区域。成功返回真，失败返回假。

### 3. “训练（）”方法

通过训练可以提高系统语音识别引擎的识别精确度，训练越多，识别系统对训练人的语音辨认越好，退出程序后，该训练结果仍然保留。“训练（）”方法执行时，系统将自动启动一个“向导”窗口，一步一步地带领用户完成语音训练任务。“训练（）”返回“真”表示训练已完成；返回“假”表示用户中止了训练。

### 4. “加入常用（）”方法

该方法在识别系统中加入常用的字词或者句子，在识别的过程中，这些字词或者句子更加容易被系统识别。该方法的执行结果仅仅对该识别程序有效，如果要重新设置常用的字词或者句子，直接再次调用该方法即可。注意，必须在识别区域被创建后能有效调用。成功返回真，失败返回假。

当计算机开始进行语音识别，获得一个识别结果的时候，需要通知程序当前识别的结果是什么。这个通知的过程，会触发“识别到语音”事件。

### 5. “识别到语音（）”方法

外部发出一段语音后，如果系统识别出当前的语音文本，则触发该事件。该事件有一个名称为“识别文本”的文本型参数，表示系统识别出的语音文本。

下面就来看看例程。

打开随书光盘本章目录中的“语音控制.e”，界面设计效果如图 23-3 所示。

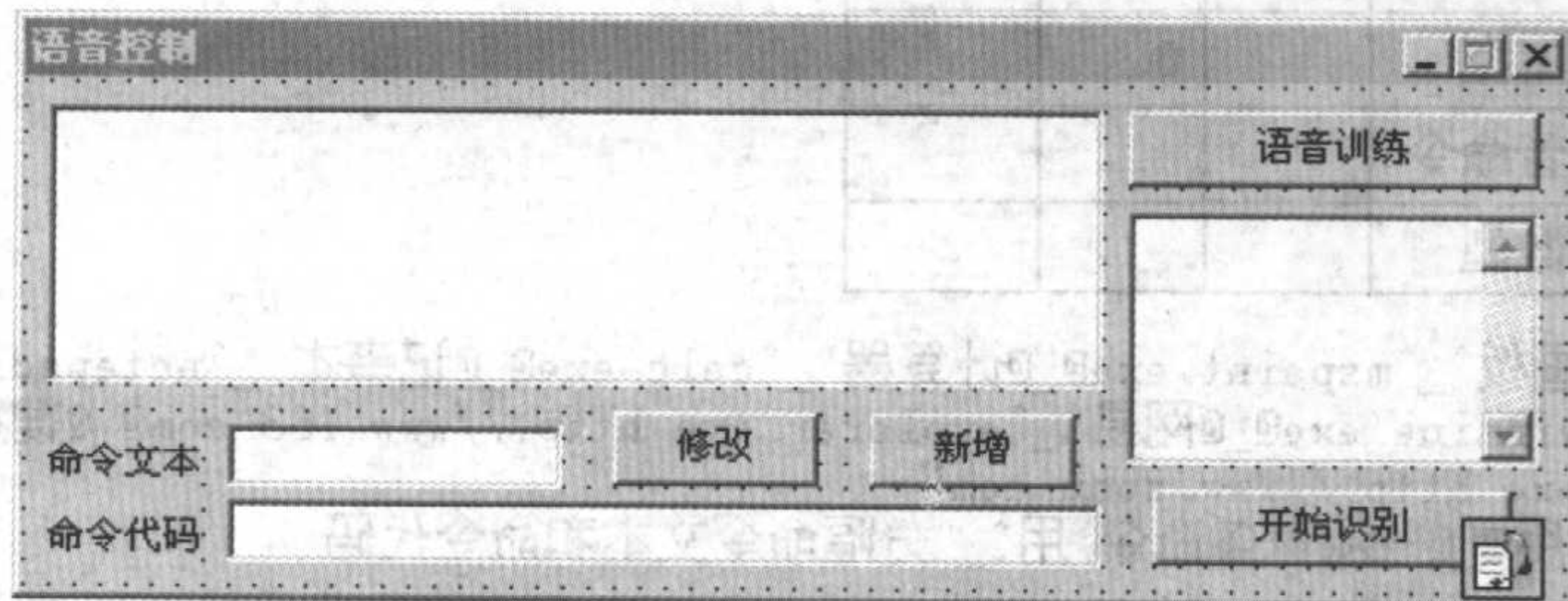


图 23-3 语音控制例程界面设计效果





语音控制程序通常分为三个部分：文本命令管理、语音训练和语音识别。例程界面左半部属于文本命令管理部分，可以新增或修改命令对应的文本和程序路径。作为例程，只是简单的启动其他外部程序，并没有对其内部进行控制。

为将命令文本与指令相互联系，程序设计了一个自定义数据类型“语音命令”，简化代码复杂程度，结构如下：

数据类型名	备注		
语音命令			
成员名	类型	数组	备注
命令文本	文本型		
命令代码	文本型		

还需要定义两个程序集变量，都是数组类型，成员数为 0。文本型数组用来保存当前指定的常用语句，语音命令型数组用来保存当前命令文本和指令。程序集变量定义如下：

窗口程序集名	备注		
窗口程序集1			
变量名	类型	数组	备注
常用语句	文本型	0	
命令	语音命令	0	

先来看看“\_\_启动窗口\_创建完毕”子程序。已定义的命令简单地使用了一个文本型变量保存，每条命令用“@\_@”分隔，命令中的命令文本和指令则用“^\_^”分隔。通过一个循环将其拆分并分别在列表框中显示，并将命令文本加入常用语句之中。

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
命令集	文本型			
命令数组	文本型		0	
临时数组	文本型		0	
临时命令	语音命令			
容器	整数型			

```
命令集 = “画板^_mspaint.exe@_@计算器^_calc.exe@_@记事本^_notepad.exe@_@扫雷^_winmine.exe@_@网易^_explorer.exe http://www.163.com@_@搜狐^_explorer.exe http://www.sohu.com”  
’ 定义命令集,用@_@分隔每条命令,用^_^分隔命令文本和命令代码  
命令数组 = 分割文本(命令集, “@_@”, )
```



```

--▶ 计次循环首 (取数组成员数 (命令数组), 容器)
  列表框1.加入项目 (命令数组 [容器], 容器)
  临时数组 = 分割文本 (命令数组 [容器], "~", )
  临时命令.命令文本 = 临时数组 [1]
  临时命令.命令代码 = 临时数组 [2]
  加入成员 (命令, 临时命令)
  加入成员 (常用语句, 临时数组 [1])
-- 计次循环尾 0
    
```

“开始识别”按钮的相关代码如下:

子程序名	返回值类型	公开	备注
_开始_被单击			

```

-- 如果 (开始.标题 = "开始识别")
  -- 如果真 (语音识别1.创建 (0, 0) = 假)
    信息框 ("启动语音识别失败", 0, )
    返回 0
  开始.标题 = "停止识别"
  语音识别1.加入常用 (常用语句)
  -- 如果真 (语音识别1.释放 (0) = 假)
    信息框 ("停止语音识别失败", 0, )
    返回 0
  开始.标题 = "开始识别"
    
```

通过对开始按钮标题的判断, 确定当前是处于语音识别状态还是停止识别状态。如果是识别状态, 则将常用语句加入。

接下来再看一下“修改”按钮的代码:

子程序名	返回值类型	公开	备注
_修改_被单击			

变量名	类型	静态	数组	备注
新命令	文本型			

```

-- 如果真 (命令文本框.内容 = "" 或 命令代码框.内容 = "" 或 列表框1.
  现行选中项 = -1)
  信息框 ("请选择列表命令", 0, )
  返回 0
  新命令 = 命令文本框.内容 + "~" + 命令代码框.内容
  -- 如果真 (信息框 ("修改:" + 列表框1.取项目文本 (列表框1.现行选中项) +
    #换行符 + "为:" + 新命令, #是否钮, ) = #否钮)
  返回 0
    
```





命令 [列表框1.取项目数值 (列表框1.现行选中项)].命令代码 = 命令代码框.内容  
 加入成员 (常用语句, 命令文本框.内容)  
 语音识别1.加入常用 (常用语句)  
 ' 防止因命令文本修改而无法识别  
 列表框1.置项目文本 (列表框1.现行选中项, 新命令)

在修改命令按钮子程序中, 为防止命令文本被改变, 除了要更改列表框显示, 还要重新将其命令文本加入常用语句中。

下面是“训练”按钮的事件处理代码:

子程序名	返回值类型	公开	备注
_训练_被单击			

--- 如果真 (语音识别1.训练 () = 假)  
 信息框 (“启动语音训练失败, 请确认已开始识别语音”, 0, )

下面请看“\_语音识别1\_识别到语音”子程序代码, 这是本例程的核心处理过程:

子程序名	返回值类型	公开	备 注		
_语音识别1_识别到语音					
参数名	类 型	参考	可空	数组	备 注
识别文本	文本型				

变量名	类型	静态	数组	备注
容器	整数型			

--- 计次循环首 (取数组成员数 (命令), 容器)  
 --- 如果真 (命令 [容器].命令文本 = 识别文本)  
   运行 (命令 [容器].命令代码, 假, )  
   编辑框1.加入文本 (识别文本 + #换行符)  
   跳出循环 ()  
 --- 计次循环尾 ()

在此处理程序中, 首先查找“命令”数组中有无匹配的“识别文本”(“识别文本”是“语音识别”组件的“识别到语音”事件的参数), 如果找到, 则运行该命令对应的程序, 同时将“识别文本”写到编辑框1的最后一行作为记录。

其余代码请读者自行查看源文件。

本程序的运行效果是, 当您对着话筒说出正确的命令(由命令文本指定)后, 相应的程序(即命令代码指定)会自动启动。



策四十二





## 第二十四章 电话语音支持库

### 24.1 支持库简介

现在很多的电话语音查询，都是服务系统控制的。用户通过按键来输入相应的查询信息，服务系统通过这些按键信息提供相应的查询服务。为了方便用户在这一服务领域的开发，易语言新增了电话语音控制支持库和一个对该支持库方法进行操作“电话控制”组件。如图 24-1 所示。

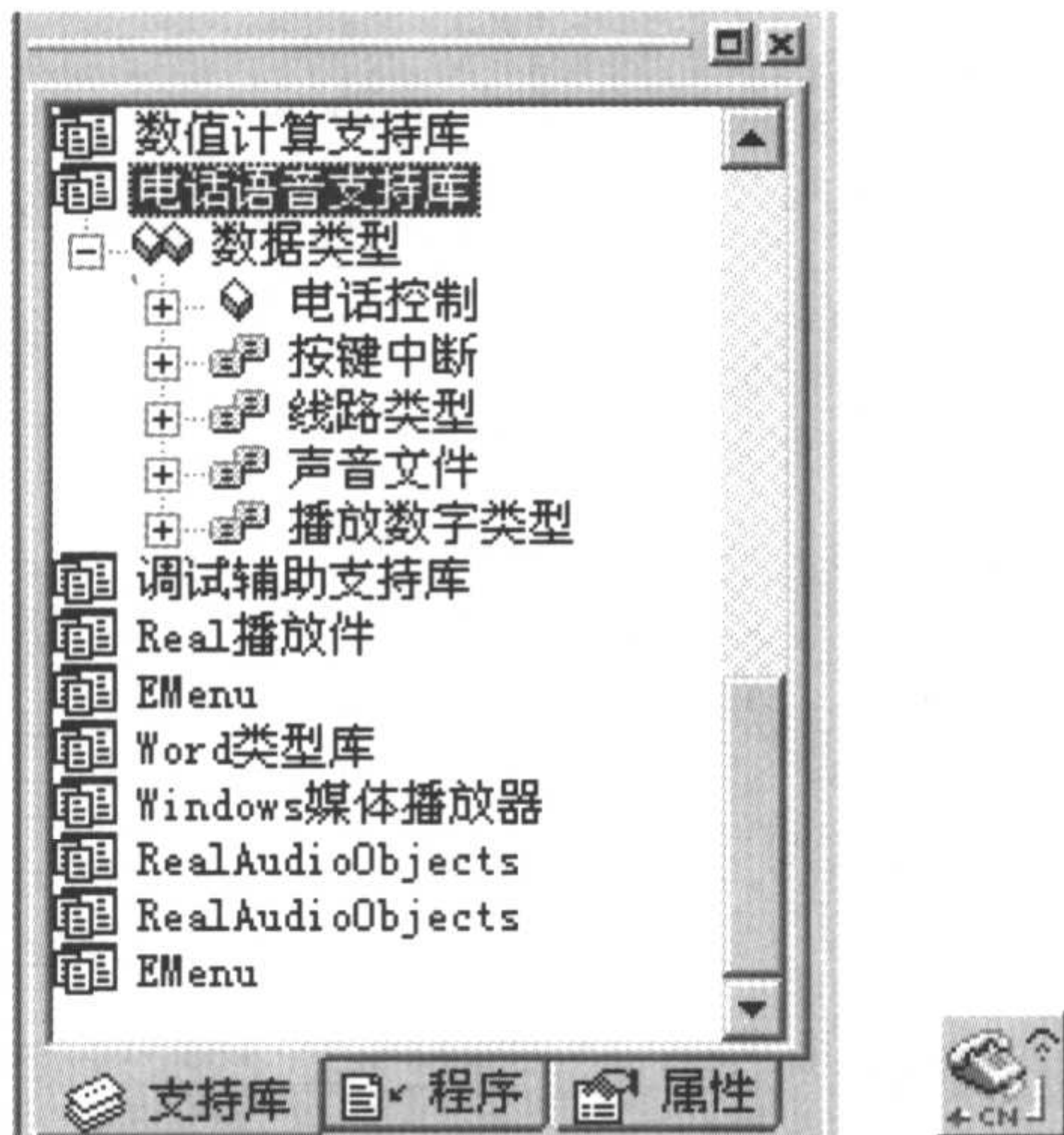


图 24-1 电话语音支持库及组件

本支持库实现了对电话的控制。需要 Windows2000 以上，DirectX9 以上版本支持。如果使用传真功能，需安装 Windows 的“传真服务”附件。语音文件必须为 WAV 格式 (PCM, 8.000kHz, 16 位, 单声道)。有些功能需要调制解调器的支持(如：语音，传真等)，需要本地网络的支持(如：来电显示等)。

### 24.2 支持库重要方法

#### 1. “呼叫 ()” 方法

〈逻辑型〉 对象. 呼叫 (终端号码, 等待时间)



用当前线路呼叫指定终端，终端类型包括电话号码，IP 地址，主机名。

“终端号码”参数，为“文本型 (text)”类型。该参数为对方的电话号码或 IP 地址或主机名。

“等待时间”参数，为“整数型 (int)”类型，该参数为要等待的时间(秒)，如果超出时间对方无应答则自动挂断。默认为 60 秒，最长等待时间是 120 秒。

## 2. “断开 ()”方法

〈整数型〉 对象. 断开 ()

断开当前与对方电话的呼叫或是连接。

## 3. “应答 ()”方法

〈逻辑型〉 对象. 应答 ()

应答对方电话的呼叫。本方法必须在“振铃”事件后调用。

## 4. “播放 ()”方法

〈逻辑型〉 对象. 播放 (文件名, 中断)

播放一个语音文件，放音完毕后会产​​生“放音完毕”事件。本命令为初级对象成员命令。

“文件名”参数，为“文本型 (text)”类型。要播放的文件名称。如果不指路径则为当前目录。

“中断”参数，为“整数型 (int)”类型。是否可以按键中断播放，可以为“按键中断”数据类型中的常量。

## 5. “录音 ()”方法

〈逻辑型〉 对象. 录音 (文件名, 中断, 时间)

录音完毕后会产​​生“录音完毕”事件。本命令为初级对象成员命令。

“文件名”参数，为“文本型 (text)”类型。存放录音文件的名称。如果不指路径则为当前目录。

“中断”参数，为“整数型 (int)”类型。是否可以按键中断播放，可以为“按键中断”数据类型中的常量。

“时间”参数，为“整数型 (int)”类型。录音的时间 (秒)。

## 6. “停止 ()”方法

〈逻辑型〉 对象. 停止 ()

停止当前的播放或录音。

## 7. “检测按键 ()”方法

〈逻辑型〉 对象. 检测按键 (是否检测按键)

开启或关闭“检测按键”事件。





“是否检测按键”参数，类型为“逻辑型 (bool)”。“真”为开启检测按键，“假”为关闭检测按键。

#### 8. “收集按键 ()”方法

〈逻辑型〉 对象. 收集按键 (按键个数, 终止按键, 按键时间 1, 按键时间 2)

收集一串按键，收集按键完毕后会生成“收集按键完毕”事件。本命令为初级对象成员命令。

“按键个数”参数，为“整数型 (int)”类型。表示要收集的按键的个数。

“终止按键”参数，为“字节型 (byte)”类型。表示终止收集的按键。

“按键时间 1”参数，为“整数型 (int)”类型。等待第一个按键所用的最长时间(秒)。

“按键时间 2”参数，为“整数型 (int)”类型。代表各按键之间的最大时间间隔(秒)。

#### 9. “初始化 ()”方法

〈逻辑型〉 对象. 初始化 ()

初始化本支持库，使用本支持库之前必须调用此方法。

#### 10. “清除 ()”方法

〈逻辑型〉 对象. 清除 ()

与初始化对应，清除本支持库所有资源。

#### 11. “设置声音文件 ()”方法

调用格式：〈逻辑型〉 对象. 设置声音文件 (声音类型, 文件名)

设置声音文件，为“播放数字 ()”方法做准备。

第 1 个参数的名称为“声音类型”，类型为“整数型 (int)”。可以为“声音文件”数据类型中的常量值。

第 2 个参数的名称为“文件名”，类型为“文本型 (text)”。本参数用来表示某个“声音类型”对应的文件名。声音文件为 WAV 格式文件。

#### 12. “播放数字 ()”方法

〈逻辑型〉 对象. 播放数字 (数字, 播放类型, 声音类型, 间隔时间)

根据“设置声音文件 ()”方法中设置的文件或以支持库默认的声音播放指定的数字。

“数字”参数，为“双精度小数型 (double)”类型。本参数用来表示要播放的数字。

“播放类型”参数，为“整数型 (int)”类型。可为“播放数字类型”数据类型中的常量值。

“声音类型”参数，为“整数型 (int)”类型。1 为 自定义; 2 为 默认女声; 3 为 默认男声; 如为 1(自定义)，要在调用此方法之前调用“设置声音文件 ()”来设置自定义声音。

“间隔时间”参数，为“整数型 (int)”类型。每播放一个数字中间间隔的时间(秒)。



## 13. “初始化传真 ( )” 方法

〈逻辑型〉 对象. 初始化传真 (设备号)

初始化传真设备。

“设备号”参数，为“整数型 (int)”类型，系统传真的设备号，默认值为 1 (是系统的第一个传真设备)。

## 14. “发送传真 ( )” 方法

〈逻辑型〉 对象. 发送传真 (电话号码, 文件名)

发送传真。

“电话号码”参数，为“文本型 (text)”类型。本参数为对方的电话号码。

“文件名”参数，为“文本型 (text)”类型。本参数为要发送的文件。

## 15. “开启接收传真 ( )” 方法

〈逻辑型〉 对象. 开启接收传真 (是否开启接收传真)

开启接收传真，当接收传真开启后有电话打过来时会自动接受传真。

“是否开启接收传真”参数，为“逻辑型 (bool)”类型。真为开启接收传真，假为关闭接收传真。

## 16. “清除传真 ( )” 方法

〈逻辑型〉 对象. 清除传真 ( )

与传真初始化对应，清除所有传真资源。

## 24.3 支持库相关例程

下面给出一个实用程序以说明本支持库的使用方法，本例程需要安装支持语音的双工调制解调器。这个例程主要实现一个简单电话语音的查询系统。因为这些功能都需要语音来实现，所以不能在书中看到效果。请大家在随书光盘的本章目录中找到该例程，自行运行来查看效果。

**注意：**在运行该例程之前，请保证运行该程序的计算机上有调制解调器、电话线连接在调制解调器上。

该电话语音查询系统，只是简单实现对电话号码的余额查询和录音的功能。所以在这里需要：一个数据库用来保存电话号码、密码和余额，10 个 “.wav” 语音文件用来存放所需要的语音提示。

本程序作了界面设计，窗口为一形象的电话机式样，按钮为图形按钮。大家可以直接打开本节例程：“电话查询系统.e”。设计时与运行时的样子如图 24-2 所示。



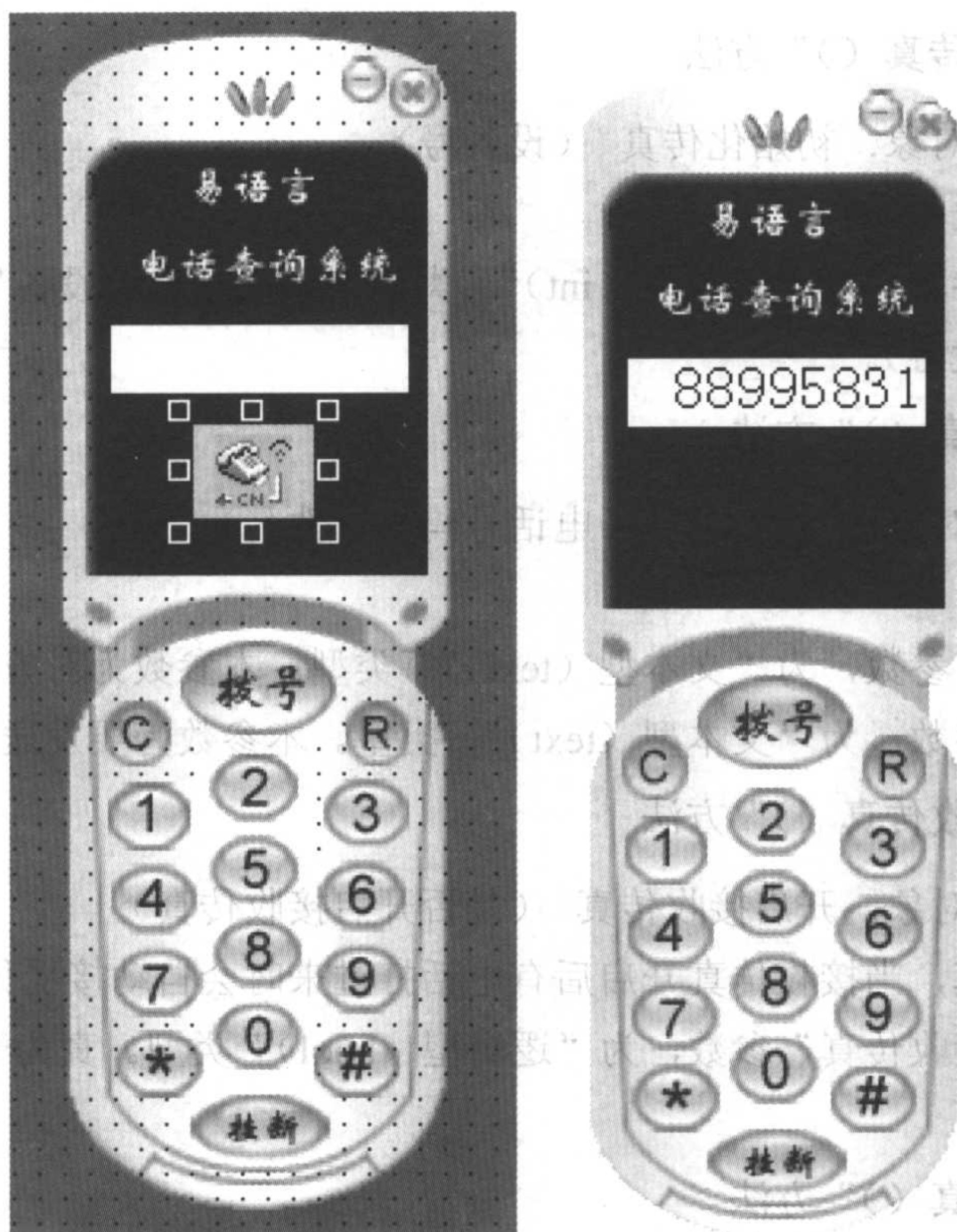


图 24-2 设计与运行界面

在程序编写之前，首先创建一些常量，以保存语音提示文件的文件名。在“\_启动窗口”的窗口程序集中创建：“录音文件名”变量，用来保存通过话筒进行录音的录音文件的文件名；“录音文件路径”变量，用来保存通过话筒进行录音的录音文件的磁盘地址；“检测按钮层数”变量，用来判断电话查询的操作步骤；“传真号码”变量，用来保存用户输入的传真号码。

要使用“电话控制”组件的方法，就必须在“\_启动窗口”创建完毕的过程中，对本支持库进行初始化。在程序结束时，用“清除（）”方法清除本支持库所有资源。在“\_\_启动窗口\_创建完毕”事件子程序中再完成一些程序的初始化任务，“检测按键层数”符值为 0，打开“余额查询数据库”，初始化传真（注意，程序结束后要用“清除传真（）”方法清除传真），在该程序运行的目录中创建一个新目录，用于保存对方电话的电话录音的 wav 文件。

“\_\_启动窗口\_创建完毕”程序内容如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
数据库	文本型			

置外形图片 (#图片1, #黑色)

电话控制.初始化 ()



```

--- 如果真 (取反 (电话控制.初始化传真 (1)))
    信息框 ("初始化传真失败", 0, )
    数据库 = "余额查询数据库.edb"
--- 如果真 (取反 (打开 (数据库, , , #禁止读写, , , )))
    信息框 ("无法打开号码数据库", 0, )
    结束 ()
检测按键层数 = 0
创建目录 (取当前目录 () + "\录音")
录音文件路径 = 取当前目录 () + "\录音\"
    
```

在电话连接之前，会有振铃事件发生，在这个事件子程序里让系统调用“应答（）”方法自动连接对方的电话，播放“欢迎语音”文件，并打开“检测按键”的功能。程序代码如下：

子程序名	返回值类型	公开	备注		
_电话控制_振铃					
参数名	类型	参考	可空	数组	备注
次数	整型				

```

--- 如果真 (次数 < 1)
    返回 ()
--- 如果真 (取反 (电话控制.应答 ()))
    信息框 (电话控制.错误信息, 0, )
    返回 ()
电话控制.播放 (#欢迎语音, #按键中断.任意键中断)
电话控制.检测按键 (真)
    
```

实现语音的查询功能，通常是通过对对方在电话机上按键，程序对对方所按键的键值来确定要进行哪一步操作。这些功能主要在“检测按键”和“收集按键完毕”的事件子程序中来实现。来一步步的进行分析：在“\_电话控制\_振铃”事件子程序中，已经启动了“检测按键”功能子的事件子程序。所以在程序自动“答应”后就会开始检测对方电话按下了哪一个按键，自然就会自动调用运行“\_电话控制\_检测按键”事件子程序。

在“\_电话控制\_检测按键”子程序中先关闭检测按键的功能，判断到“检测按键层数”为 0 后，符值“检测按键层数”为 1，来准备下一次进行按键收集后判断所要做的操作。“\_电话控制\_检测按键”子程序中所返回的“键值”是电话机上的所按键的键值，根据这一点来判断下一步要进行的操作，操作包括进行电话余额查询和电话录音；其中键值 49 代表“数字键 1”，按下后会进行语音查询功能，键值 50 代表“数字键 2”，按下后会进行电话录音。部分程序代码如下：

子程序名	返回值类型	公开	备注		
_电话控制_检测按键					
参数名	类型	参考	可空	数组	备注
键值	字节型				





电话控制.检测按键 (假)

```
--- 判断 (检测按键层数 = 0)
    检测按键层数 = 1
        --- 判断 (键值 = 49)
            电话控制.播放 (#输入电话号码语音, #按键中断.不可中断)
        --- 电话控制.收集按键 (20, 35, 20, 20)
            ▶ 判断 (键值 = 50)
                --- 电话控制.播放 (#录音提示, #按键中断.不可中断)
```

如果按下“数字键 1”就可进行电话余额的查询操作了,播放电话提示音提示输入电话号码,“收集按键 ()”方法进行按下按键收集工作,在完成收集后便进入了“\_电话控制\_收集按键完毕”事件子程序中,在此提供了所收集到的“按键串”。判断“检测按键层数”值为 1 进行查询操作,把该变量符值为 2,查找数据库中的电话号码有没有与按键串相等的,如果相等就提示输入密码,再次“收集按键”。

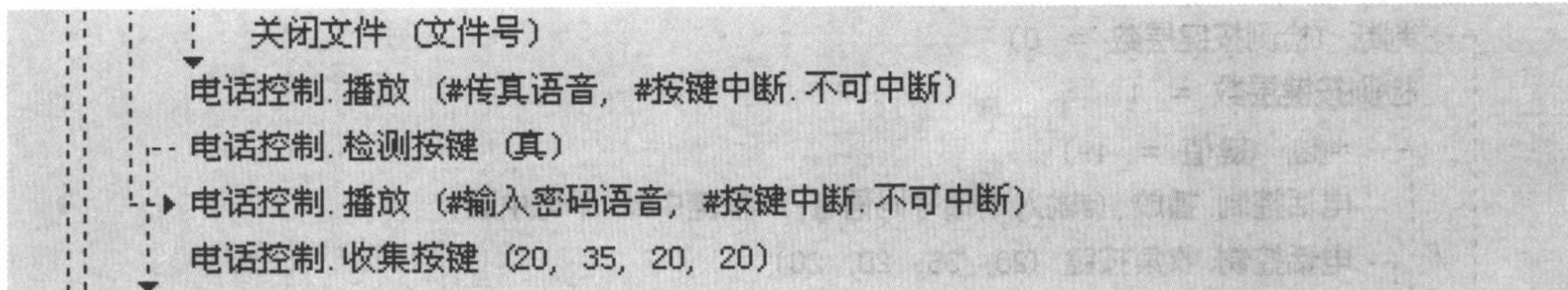
“\_电话控制\_收集按键完毕”子程序部分程序代码如下:

```
--- 判断 (检测按键层数 = 1)
    --- 如果 (查找 (读 (“电话号码”) = 按键串))
        号码 = 按键串
        检测按键层数 = 2
        电话控制.播放 (#输入密码语音, #按键中断.不可中断)
    --- 电话控制.收集按键 (20, 35, 20, 20)
        ▶ 电话控制.播放 (#输入电话号码语音, #按键中断.不可中断)
        电话控制.收集按键 (20, 35, 20, 20)
```

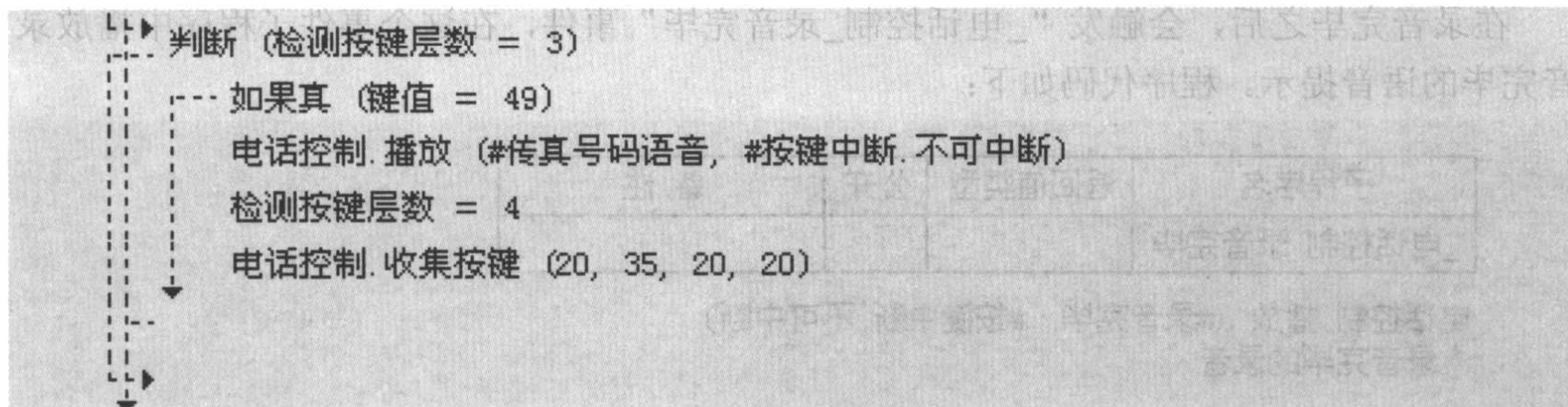
“电话密码”收集按键完成,再次进入“\_电话控制\_收集按键完毕”事件子程序中,判断“检测按键层数”值为 2,如果判断成功改值为 3。进入密码判断当中,如果密码正确播放当前记录行中的余额,并新建一个“余额.txt”文件 (“传真临时文件”常量中的值),并提示进行传真传送这条记录,开启“检测按键”功能再次进行按键检测。代码如下:

```
▶ 判断 (检测按键层数 = 2)
    --- 如果 (读 (“密码”) = 按键串)
        检测按键层数 = 3
        当前余额 = 读 (“余额”)
        电话控制.播放 (#余额语音, #按键中断.不可中断)
        电话控制.播放数字 (当前余额, #播放数字类型.金额, 3, 2)
        文件号 = 打开文件 (#传真临时文件, #重写, #无限制)
        --- 如果真 (文件号 ≠ 0)
            写出文本 (文件号, 号码 + “ ” + 按键串 + “ ” + 到文本 (当前余额))
```

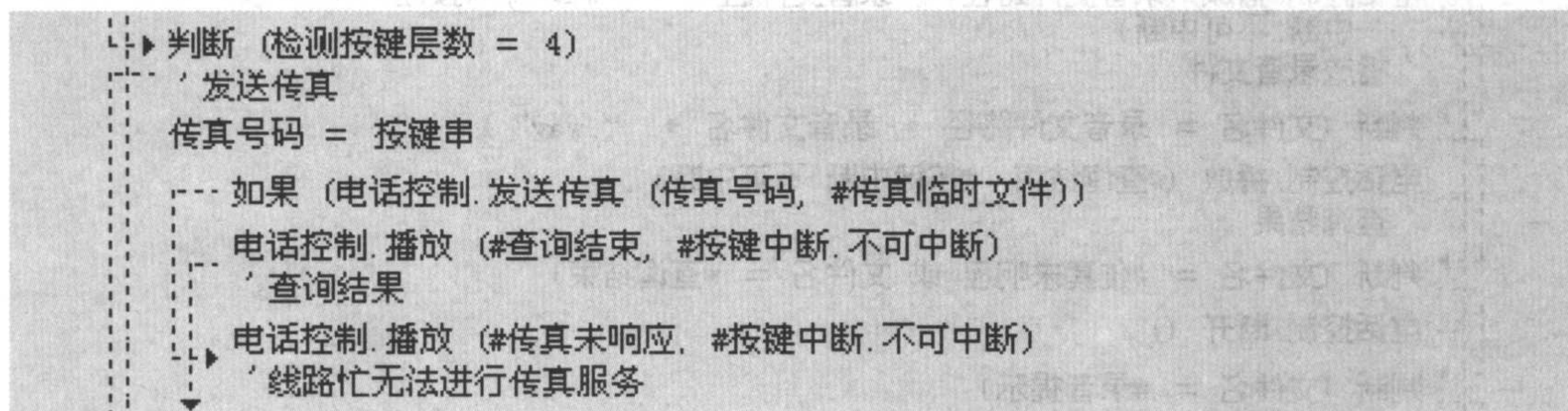




再次进行按键检测，进入“\_电话控制\_检测按键”事件子程序中，关闭“检测按键”功能。判断“检测按键层数”值为 3，判断是否按下的“数字键 1”（键值为 49），如果真就提示输入传真号码，将“检测按键层数”为 4，最后一次收集对方电话机按键。



最后收集电话机按键完成，进行“\_电话控制\_收集按键完毕”事件子程序。判断“检测按键层数”为 4 后，发送传真。如果所发传真未响应，提示传真未响应语音，否则提示查询结束，并在“\_电话控制\_播放完毕”事件子程序里面断开电话连接——这步操作要在“\_电话控制\_播放完毕”事件子程序中完成。



上面所讲的是电话查询功能的程序流程，下面再来看一下电话录音的电话流程。

在欢迎语音之后，按下电话机上的“数字 2”键，进入“\_电话控制\_检测按键”事件子程序，判断“检测按键层数”为 0，将值该变量为 1。判断键值为 50（是否按下电话机上的“数字 2”键），播放录音提示文件，在此文件播放完毕之后会触发“\_电话控制\_播放完毕”事件，在该事件子程序中实现录音操作。程序代码如下：

子程序名	返回值类型	公开	备注		
_电话控制_检测按键					
参数名	类型	参考	可空	数组	备注
键值	字节型				

电话控制. 检测按键 (假)





```
判断 (检测按键层数 = 0)
检测按键层数 = 1
判断 (键值 = 49)
电话控制.播放 (#输入电话号码语音, #按键中断.不可中断)
电话控制.收集按键 (20, 35, 20, 20)
判断 (键值 = 50)
电话控制.播放 (#录音提示, #按键中断.不可中断)
```

在录音完毕之后，会触发“\_电话控制\_录音完毕”事件，在这个事件子程序中播放录音完毕的语音提示。程序代码如下：

子程序名	返回值类型	公开	备注
_电话控制_录音完毕			

电话控制.播放 (#录音完毕, #按键中断.不可中断)  
'录音完毕的录音

当录音完毕的语音提示播放完毕之后，播放刚刚录过的电话录音。这步操作要在“\_电话控制\_播放完毕”件事中完成。下面就是“\_电话控制\_播放完毕”事件子程序中的代码：

```
判断 (文件名 = #录音完毕)
电话控制.播放 (录音文件路径 + 录音文件名 + “.wav”, #按键中断.不可中断)
'播放录音文件
判断 (文件名 = 录音文件路径 + 录音文件名 + “.wav”)
电话控制.播放 (#查询结束, #按键中断.不可中断)
'查询结果
判断 (文件名 = #传真未响应 或 文件名 = #查询结束)
电话控制.断开 ()
判断 (文件名 = #录音提示)
录音文件名 = 时间到文本 (取现行时间 (), )
如果真 (取反 (电话控制.录音 (录音文件路径 + 录音文件名 + “.wav”, #按键中断.不可中断, 10)))
信息框 (电话控制.错误信息, 0, )
```

在这个程序中，也可以拨打和挂断电话，只需要“呼叫 ()”和“断开 ()”两方法即可。

“\_呼叫钮\_被单击”事件子程序：



子程序名	返回值类型	公开	备注
_呼叫钮_被单击			

如果真 (取反 (电话控制.呼叫 (电话号码框.内容, 60)))  
信息框 (电话控制.错误信息, 0, )

“\_断开钮\_被单击”事件子程序如下:

子程序名	返回值类型	公开	备注
_断开钮_被单击			

电话控制.断开 ()

在对方来电话时,也可以显示出对方电话的电话号码,该功能要在“\_电话控制\_来电显示”事件子程序中实现。程序代码如下:

子程序名	返回值类型	公开	备注		
_电话控制_来电显示					
参数名	类型	参考	可空	数组	备注
来电号码	文本型				

电话号码框.内容 = 来电号码

结束程序编写,可以进行运行测试,来查看运行效果。





## 第二十五章 数码设备支持库

### 25.1 支持库简介

“数码设备”支持库可以对所需要的图片进行动态的抓取，可通过摄像头的“视频捕获照片”窗口来捕获图像。而且该支持库还支持扫描仪设置，可对图片进行静态的描扫。如图 25-1 所示，是数码设备支持库得数据类型显示。

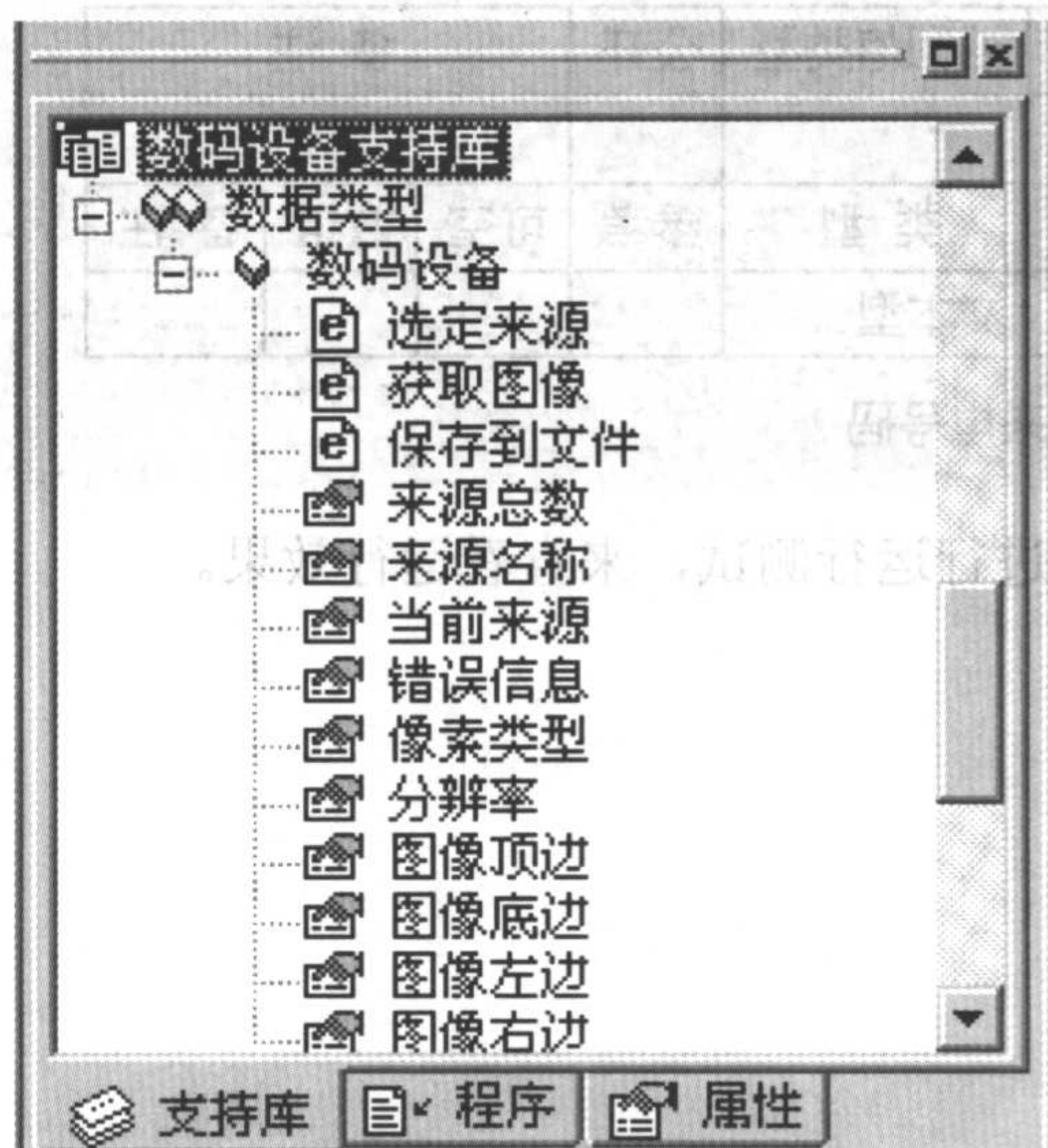


图 25-1 数码设备支持库

### 25.2 支持库属性与方法

#### 25.2.1 数码设备的重要属性

1. “来源总数”属性  
返回系统中可用的数码设备的总数。
2. “来源名称”属性  
返回当前使用的数码设备的名称。
3. “当前来源”属性



设置或返回当前数码设备号, 范围从 0 至来源总数。

#### 4. “像素类型”属性

设置或返回当前数码设备的像素类型; 0: 黑白, 1: GRAY, 2: RGB, 3: PALETTE, 4: CMY, 5: CMYK, 6: YUV, 7: YUVK, 8: CIEXYZ。

#### 5. “横向分辨率”属性

数据类型为: 整数型。设置或返回横向分辨率 (dpi)。

#### 6. “纵向分辨率”属性

数据类型为: 整数型。设置或返回纵向分辨率 (dpi)。

#### 7. “图像顶边”、“图像底边”、“图像左边”、“图像右边”属性

分别设置或返回设备中图像的顶边、底边、左边和右边, 单位为英寸。

**注意:** “像素类型”、“分辨率”、“图像顶边”、“图像底边”、“图像左边”和“图像右边”属性主要用来设置扫描仪的预览窗口中扫描边界的设定, 有些摄像头设备的厂商并没有设置对这些属性支持。

### 25.2.2 数码设备的方法

#### 1. “初始化 ()”方法

〈逻辑型〉 对象. 初始化 ()

用于初始化数码设备支持库。**注意:** 对数码设备支持库进行初始化之后, 才可以使用该支持库中的相关命令。

#### 2. “选定来源 ()”方法

〈整数型〉 对象. 选定来源 (是否用默认选择, 来源选项)

用于设置选择用哪个设备来获取图像, 返回值为 1, 表示选取了新的来源; 返回值为 2, 表示取消了选择; 返回值为 -1 表示出错。

“是否用默认选择”参数, 为“逻辑型 (bool)”类型。本参数表示是否使用系统默认选择。

“来源选项”参数, 为“整数型 (int)”类型, 初始值为“-1”。当此参数为 -1 时, 会弹出默认的对话框让用户进行设备来源选择; 0 至“来源总数”表示选择的设备来源号。

#### 3. “获取图像 ()”方法

〈逻辑型〉 对象. 获取图像 (是否显示用户界面)

用于从所选的设备中获取图像, 如果获取成功会触发图片传送完毕事件。

“是否显示用户界面”参数, 为“逻辑型 (bool)”类型。该参数表示是否显示用户界面。例如: 如果该参数为真, 用摄像头设备的“视频捕获照片”窗口来捕获图像; 否则不显示“视频捕获照片”窗口, 直接对图像进行捕获。

#### 4. “保存到文件 ()”方法

〈逻辑型〉 对象. 保存到文件 (文件名)

将捕获到的图片保存到文件中, 图片格式为 BMP 格式。**注意:** 此方法需要在“图片发送完毕”事件中调用。

“文件名”参数, 为“文本型 (text)”类型。用于提供要保存文件的文件名。





## 25.3 支持库相关例程

了解“数码设备”支持的相关属性和方法后，运用该支持库的相关命令来做一个简单的大头贴程序，如图 25-2 所示，这是大头贴程序运行后的界面效果。在这个程序里面主要实现数码摄像、变换花框样式和保存、打印制作完成后的大头贴图像。



图 25-2 运行后效果

大头贴是一种为手指样大小的贴纸，可贴在书本、铅笔盒和手机等多种物品的表面上。大头贴有不同大小的尺寸，可在调选时调节任意大小进行打印。

这个程序主要是通过数码摄像头来获取一张数码照片，选择不同的花框为照片添加小小的样式花边。可以把含花框的叠加照片保存成 BMP 格式的图片，也可以通过打印机打印出来。

先来做一些程序制作之前的准备工作：八张照片的花框图片，和一个数码摄像头。

首先新建一个易程序，在“\_启动窗口”中添加组件，如图 25-3 所示：

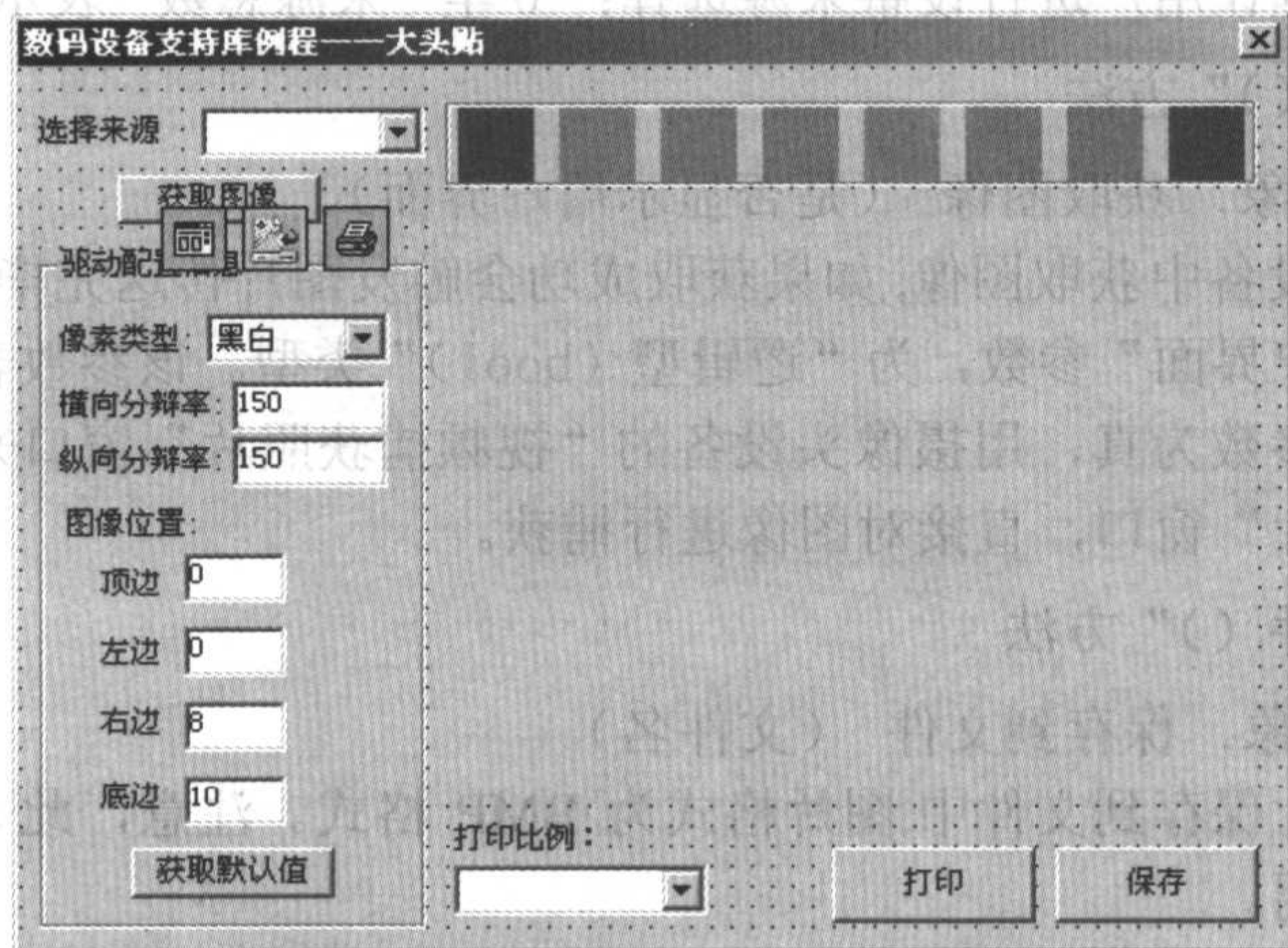


图 25-3 添加组件



**组合框:**

名称分别为: 设备来源名称框、图像比例框、像素类型框

**按钮:**

名称分别为: 获取图像钮、获取默认值钮、打印钮、保存钮

**编辑框:**

名称分别为: 横向分辨率、纵向分辨率、图像顶边、图像左边、图像右边、图像底边

**画板:**

名称为: 照片显示板

还需要添加: 一个“数码设备”组件, 用于进行数码设备的相关设置; 一个“通用对话框”组件, 用于提取保存文件名; 一个“打印机”组件, 用于打印设置后的照片。八个图片框, 用于添加八个照片花框的缩略图;

把这个八个照片花框图片, 作为图片资源保存在易语言的“资源表”中。名称分别为“图片1”至“图片8”。

创建一个数值型程序集变量: “照片号”, 用于保存摄像头所拍图片的图片号; 再创建一个数值型程序集数组: “图片号数组”, 用来保存八个照片花框图片的图片号。

程序运行时: 首先用数码设备支持库进行初始化, 系统会自动查找数码设备, 并把这些设备的名称显示在“设备来源名称框”中, 以供用户对数码设备进行相应选择; 图片框应显示照片花框的缩略图; 并且把八个图片的图片号放入一个名为“图片号数组”(程序集数组)当中。这些代码可以写在“\_启动窗口\_创建完毕”事件子程序当中(在这里还加入了一个XP窗口风格)。

在这里要注意, “数码设备”的“当前来源”属性是从0开始的, 而“计次循环首”是从1开始进行循环。所以, 在循环添加“设备来源名称框”中的项目的时候, 要先把“数码设备”的“当前来源”减1。下面是“\_启动窗口\_创建完毕”事件子程序的部分代码:

变量名	类型	静态	数组	备注
循环次数	整数型			

XP风格 (2)

数码设备. 初始化 0

--- 计次循环首 (数码设备. 来源总数, 循环次数)

    数码设备. 当前来源 = 循环次数 - 1

    设备来源名称框. 加入项目 (数码设备. 来源名称, )

--- 计次循环尾 0

图片框1. 图片 = #图片1

图片框2. 图片 = #图片2

图片框3. 图片 = #图片3

图片框4. 图片 = #图片4

图片框5. 图片 = #图片5

图片框6. 图片 = #图片6

图片框7. 图片 = #图片7





图片框8.图片 = #图片8

图片号数组 [1] = 载入图片 (#图片1)

图片号数组 [2] = 载入图片 (#图片2)

运行完毕之后,用户就可以在“设备来源名称框”中选择所需要的数码设备名称,在这里请用户选择摄像头设备名称。并按下“获取图像钮”来打开摄像头的“视频捕获照片”窗口来进行拍照,并进行捕获图像的操作,显示图像的四边边界位置。实现这些功能的代码可以写在“\_获取图像钮\_被单击”事件子程序中。

**注意:**在这里,数码设备的“选定来源()”方法的“是否用默认选择”参数设置为“假”,“来源选项”设置成为“设备来源名称框”(组合框)的现行选中项。如果用户没有在“设备来源名称框”中对数码设备进行选择,那么它的“现行选中项”值就为默认值-1。而在“选定来源()”方法的“来源选项”参数的默认值-1时,就弹出默认的对话框让用户选择。所以这样就添加了一个很好的系统维护功能。

子程序名	返回值类型	公开	备注
_获取图像钮_被单击			

```
--- 如果真 (数码设备.选定来源 (假, 设备来源名称框.现行选中项) = 1)
    设备来源名称框.现行选中项 = 数码设备.当前来源
    数码设备.像素类型 = 像素类型框.现行选中项
    数码设备.横向分辨率 = 到数值 (横分辨率.内容)
    数码设备.纵向分辨率 = 到数值 (纵分辨率.内容)
    数码设备.图像顶边 = 到数值 (图像顶边.内容)
    数码设备.图像底边 = 到数值 (图像底边.内容)
    数码设备.图像左边 = 到数值 (图像左边.内容)
    数码设备.图像右边 = 到数值 (图像右边.内容)
    数码设备.获取图像 (假)
```

从所选的设备中获取图像成功之后,会触发“图片传送完毕”事件。在“\_数码设备\_图片传送完毕”事件的子程序中,使图片显示在“照片显示板”中,并把所得到图片的图片号赋值给程序集变量“照片号”。最后用“保存到文件()”方法将照片保存到该程序的运行目录中。通过新建的静态变量“图片数”的不断累加1来动态添加文件名称。

子程序名	返回值类型	公开	备 注		
_数码设备_图片传送完毕					
参数名	类 型	参考	可空	数组	备 注
图片	字节集				
图片宽度	整数型				
图片高度	整数型				

变量名	类型	静态	数组	备注
图片数	整数型	✓		



```

卸载图片 (照片号)
照片号 = 载入图片 (图片)
照片显示板.画图片 (照片号, 0, 0, , , #拷贝)
数码设备.保存到文件 (取运行目录 () + "\" + 到文本 (图片数) + ".BMP")
图片数 = 图片数 + 1
    
```

拍完数码照片之后, 点击花框样式缩略图就可为该照片选择喜欢的花框样式了。在这里要说明一下, 照片花框图是如何添加在照片上的, 是用画板的“画图片 ( )”方法, 分别把照片和花框样式画到画板当中。花框中间区域的颜色是纯白色的, 为了显示下面照片, 在“画图片 ( )”方法的“图片画出方法”参数中把花框样式图片中间的纯白色减去, 以达到指定透明色的效果。

照片显示板的“清除”方法, 用于在选择不同的花框图片时进行更新重画图片。这段代码可以反复的写在八个“图片框”的“鼠标左键被按下”事件子程序中。

子程序名	返回值类型	公开	备注		
_图片框1_鼠标左键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

```

照片显示板.清除 ( , , )
照片显示板.画图片 (照片号, 0, 0, , , #拷贝)
照片显示板.画图片 (图片号数组 [1], 0, 0, , , -取颜色值 (255, 255, 255))
    
```

选择照片设置花框样式之后, 就可以对其进行打印和保存了。用户也可以设置打印的图片比例。下面就是“保存钮”和“打印钮”被单击后所进行的代码流程。

子程序名	返回值类型	公开	备注
_保存钮_被单击			

```

通用对话框1.类型 = 1
通用对话框1.过滤器 = "BMP (*.BMP;*.RLE;*.DIB)|*.BMP"
--- 如果真 (通用对话框1.打开 () = 真)
    ↓
    写到文件 (通用对话框1.文件名, 照片显示板.取图片 ( ))
    
```

在“\_打印钮\_被单击”事件子程序中, 在打印机“画图片”方法的“图片画出宽度”和“图片画出高度”参数中对图片的比例大小进行设定。





子程序名	返回值类型	公开	备注
_打印钮_被单击			

打印机1.开始打印 (真, 真, , )

打印机1.画图片 (载入图片 (照片显示板.取图片 ( )), 0, 0, 照片显示板.取图片宽度 (载入图片 (照片显示板.取图片 ( ))) × 到数值 (图像比例框.内容) ÷ 100, 照片显示板.取图片高度 (载入图片 (照片显示板.取图片 ( ))) × 到数值 (图像比例框.内容) ÷ 100, )

打印机1.结束打印 ()

结束程序编写，可以进行运行测试，来查看运行效果，如图 25-2 所示。

该程序也可以选择相应的扫描仪进行图片扫描，所得图片会显示在照片显示板中，如图 25-3 所示。



图 25-3 扫描界面

以上例程名称为：“大头贴.e”。

## 25.4 视频设备

数码设备支持库最新版本又增加对动态视频的支持，大家可以看到支持库列表中又增加了一个新“视频设备”数据类型。包含 12 个成员命令，12 个成员属性，1 个组件事件。

下面大家可以打开另一个例程：“动态视频.e”。这个例程可支持数码摄像头动态的影像播放。界面如图 25-4 所示。



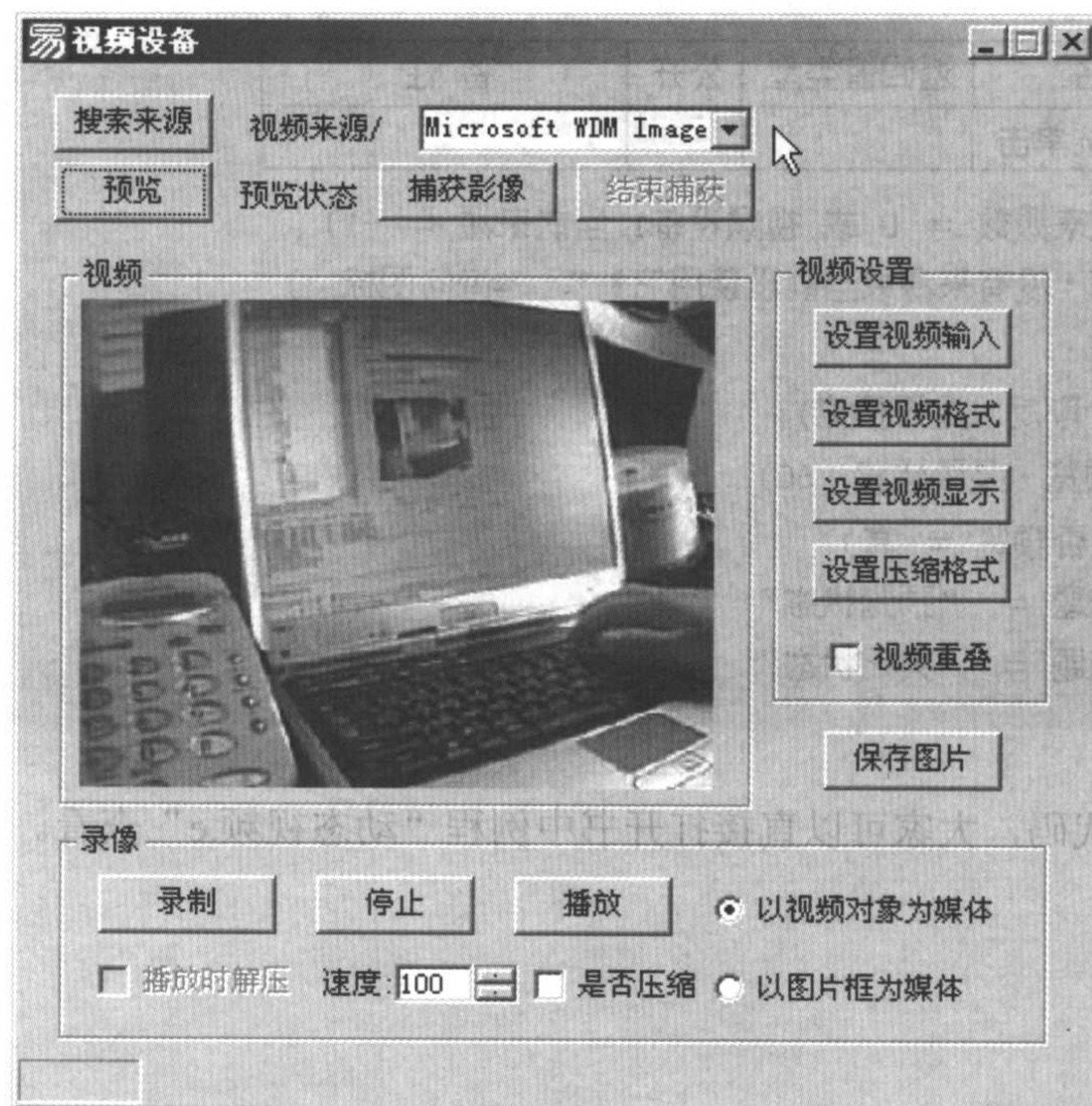


图 25-4 程序界面

本程序同样也是先将已安装的数码设备保存在列表中，供用户选择，再点击“预览”按钮就可以显示数码摄像头中的动态影像了。

其中“获取来源”用户自定义子程序如下：

子程序名	返回值类型	公开	备注
获取来源			

变量名	类型	静态	数组	备注
计数器	整数型			

```

--> 计次循环首 (组合框2.取项目数 (), 计数器)
    视频设备1.当前来源 = 计数器
    组合框2.置项目文本 (计数器 - 1, 视频设备1.来源名称)
--- 计次循环尾 ()
--> 变量循环首 (组合框2.取项目数 () + 1, 来源数, 1, 计数器)
    视频设备1.当前来源 = 计数器
    组合框2.加入项目 (视频设备1.来源名称, 计数器)
--- 变量循环尾 ()
初始化 = 1
    
```

“预览”按钮的子程序如下：





子程序名	返回值类型	公开	备注
预览按钮_被单击			

```

--- 如果真 (来源数 = 0 或 视频设备1.当前来源 = -1)
    信息框 ("没有来源或当前没使用它!", #询问图标, )
    返回 0
是否预览 = 取反 (是否预览)
视频设备1.预览 (是否预览, 60)
--- 判断 (是否预览 = 真)
    标签3.标题 = "预览状态"
--- 标签3.标题 = "关闭状态"

```

其他的程序代码，大家可以直接打开书中例程“动态视频.e”查看。



图 10-1-1 动态视频

“动态”是指，在播放过程中，可以随时暂停、继续、快进、后退、静音、音量、全屏、退出等操作。

在播放过程中，可以随时暂停、继续、快进、后退、静音、音量、全屏、退出等操作。

在播放过程中，可以随时暂停、继续、快进、后退、静音、音量、全屏、退出等操作。

主窗	公共	私有	私有
			私有

变量	数据类型	初始值
变量1	字符串	字符串

(播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出)

播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

(播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出)

0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

(播放时，0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出)

0 表示暂停，1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

1 表示播放，2 表示快进，3 表示后退，4 表示静音，5 表示音量，6 表示全屏，7 表示退出

在播放过程中，可以随时暂停、继续、快进、后退、静音、音量、全屏、退出等操作。



## 第二十六章 脚本语言支持组件

### 26.1 组件简介

Windows Script 是一种 Windows 平台下的综合脚本底部结构。Windows Script 提供的脚本引擎、Visual Basic Scripting Edition 和 Microsoft JScript, 可以被嵌入到 Windows 应用程序中, 并可帮助用户为 Windows 应用程序编写脚本时进行调试。

易语言所提供的“脚本组件”能够借助 Windows Script 解析执行 VBScript 和 JScript 两种类型的脚本。该组件除了作为脚本调试工具以外, 还可以借助其执行和运算功能为程序添加一些额外的功能, 比如插件功能等。

脚本文件编写十分简单, 可实现部分系统基本操作, 如磁盘文件操作、注册表操作、外部文件调用、系统设置等等。本章主要讲解组件功能的使用, 并不涉及脚本的具体编写。如果要研究脚本编程, 建议在阅读本章内容之后, 自行查阅相关资料。

### 26.2 属性和方法

“脚本组件”的属性和方法不多, 使用起来相当简便。

#### 1. “语言”属性

设置(或返回)即将使用(或正在使用)的脚本语言名称, 可设为 JScript 或 VBScript。不同的脚本语言对应不同的语法规则, 不可混用。

#### 2. “错误信息”属性

返回当前脚本执行过程中的错误信息。若错误信息提示缺少“msscript.ocx”, 请将随书光盘本章例程目录中的 msscript.ocx 文件复制到系统 System 目录中, 选择屏幕左下角“开始”→“运行”, 输入 regsvr32 c:\windows\system\msscript.ocx, 点击确定按钮, 即可解决此问题。

#### 3. “超时”属性

设置运行中断间隔时间(毫秒)。当脚本执行超过此时间后, 会弹出系统对话框提示需要继续执行或结束执行, 避免在调试过程中遇到死循环函数。

#### 4. “执行( )”方法

加载指定的代码文本并自动执行。返回真为执行正常, 返回假为出错, 错误信息可以



从“错误信息”属性中获取。

#### 5. “运行 ()”方法

运行代码文本中指定的函数或过程，此代码文本必须经“执行”方法预先加载。如有返回值就返回相关数据，否则返回空文本。“执行 ()”方法没有显示返回信息，可用此方法对单个函数进行调试。

#### 6. “清除 ()”方法

清除所加载的代码文本。

#### 7. “计算表达式 ()”方法

可实现对形如  $1+(2*3)/6+10-10$  的四则表达式进行混合运算并返回结果。

## 26.3 组件应用实例

### 26.3.1 四则表达式计算器

计算四则表达式运算结果常用的方法是：用两个数组做堆栈，然后根据运算符的优先级来进行判断处理，非常麻烦。现在可以借助脚本语言组件的“计算表达式 ()”方法取得四则表达式运算结果，程序中实现计算部分仅用了一句代码。

下面给出一个四则表达式计算器例程，以说明“脚本组件”的“计算表达式 ()”方法 的用法。

打开随书光盘本章例程目录中的“四则表达式计算器.e”，设计界面和运行界面如图 26-1 所示。

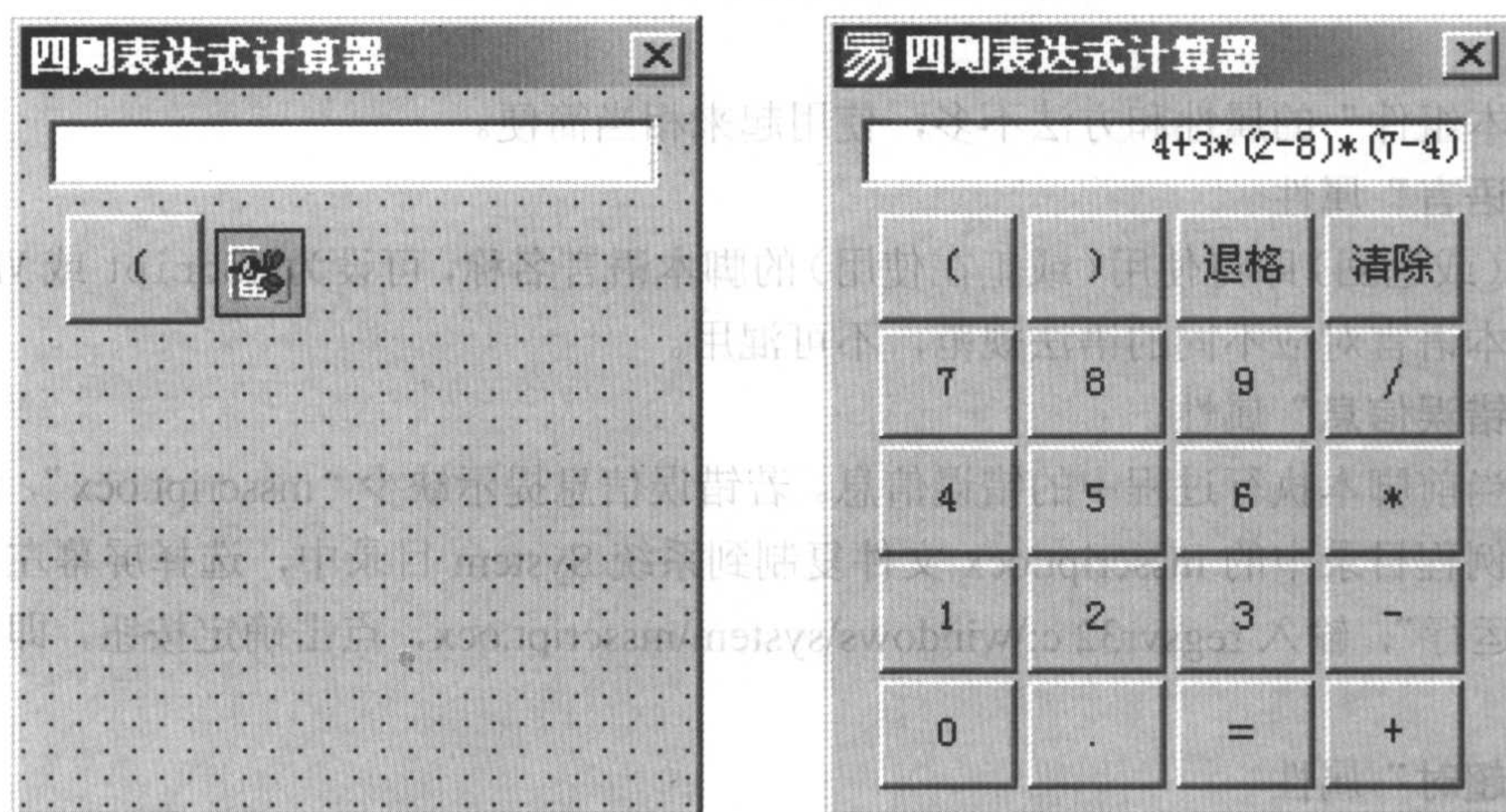


图 26-1 四则表达式计算器设计界面与运行界面

程序使用一个编辑框用于输入表达式。为了简化代码复杂程度，采用了动态创建窗口组件的方法生产所有的数字或运算符按钮，因此窗体上只放了一个按钮，大小设为  $40 \times 32$



像素。其他的按钮将根据第一个按钮的尺寸和位置来动态创建。窗体上放了一个脚本语言组件，用来进行四则表达式运算处理。

首先，为动态生成的按钮组件创建一个按钮数组。通过循环方式来完成对所有按钮的定位、赋标题值和显示操作。代码如下：

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

变量名	类型	静态	数组	备注
按键	按钮		19	
容器	整数型			

```

--> 计次循环首 (19, 容器)
    复制窗口组件 (主按键, 按键 [容器])
    按键 [容器].左边 = 主按键.左边 + (容器 % 4) × 43
    按键 [容器].顶边 = 主按键.顶边 + (容器 \ 4) × 34
    按键 [容器].标题 = 多项选择 (容器, ")", "退格", "清除", "7", "8",
        "9", "/", "4", "5", "6", "*", "1", "2", "3", "-",
        "0", ".", "=", "+")
    按键 [容器].可视 = 真
-- 计次循环尾 ()
编辑框1.获取焦点 ()
    
```

以上代码中，使用了“复制窗口组件（）”命令，以“主按键”按钮为基础，复制出另外 19 个按钮，调整各自的位置和标题，以动态地生成程序操作界面。需要提醒大家的是，使用“复制窗口组件（）”复制出来的组件，其事件将被自动转交到“复制源组件（此处为‘主按键’）”，这样非常便于处理——不必为每一个按钮都单独写一个事件处理子程序了。

此时，可以按 F5 运行程序看看效果，现已经具备计算器的雏形了。剩下的工作就是对输入表达式的处理，以及运算的操作了。

根据被单击按钮的标题来做判断处理。考虑“退格”、“清除”、“=”和普通数字（运算符）四种情况。运算部分仅一行代码就解决了问题：

```
编辑框1.内容 = 脚本组件1.计算表达式 (编辑框1.内容)
```

光标插入位置的判断代码并不复杂，请读者自行理解。

至此，整个编程工作就全部完成了，可以运行程序测试一下计算结果是否正确。

### 26.3.2 外部程序调用

通过“脚本语言”支持组件执行脚本文件，可以直接运行外部程序，或通过 COM 接口对外部程序进行操作。

打开随书光盘本章例题目录中的“脚本演示.e”。运行效果如图 26-2 所示。



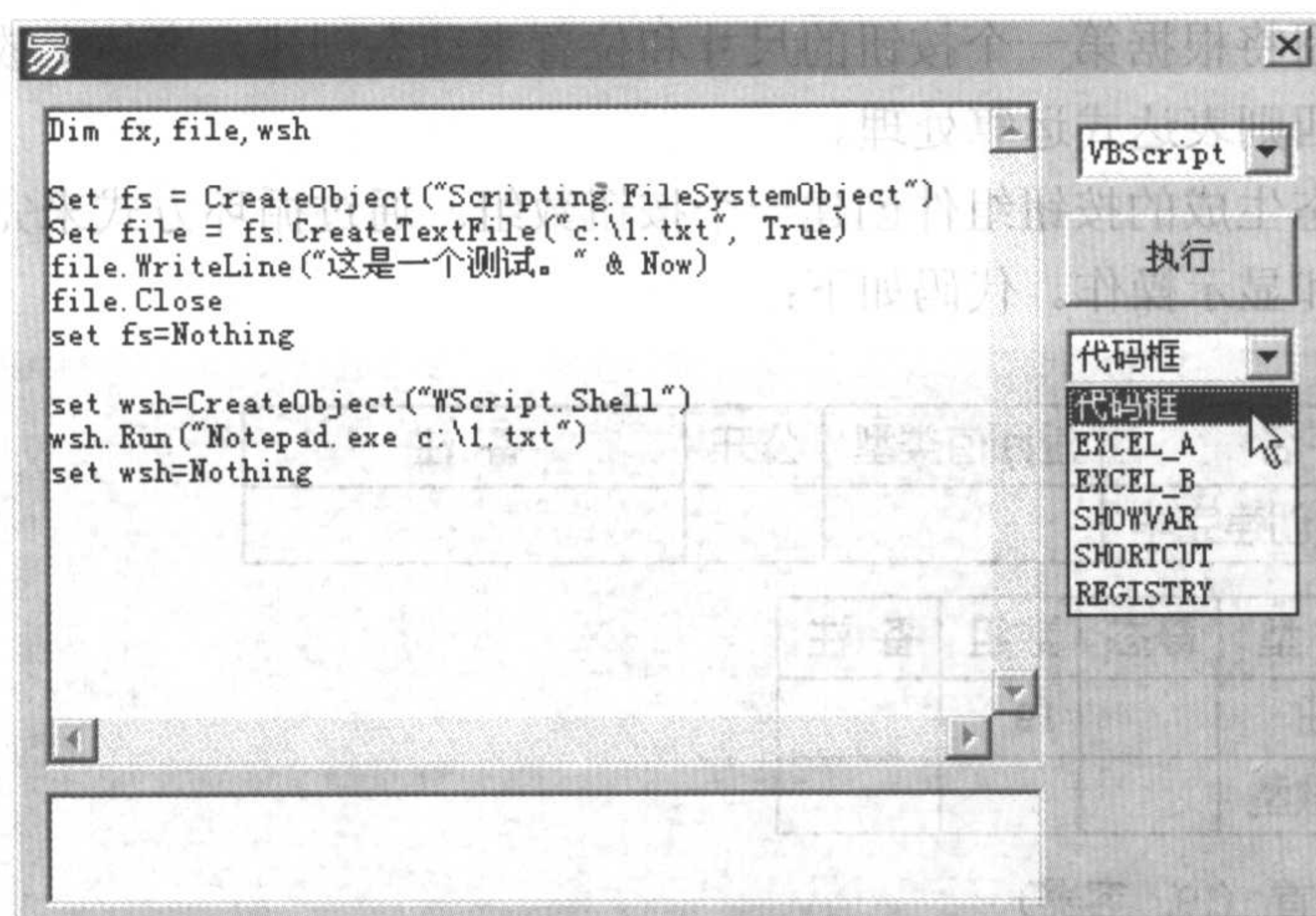


图 26-2 执行脚本代码

这个例程演示了如何使用“脚本语言”组件执行脚本代码，调用记事本、EXCEL 等程序。核心代码为：

子程序名	返回值类型	公开	备注
执行_被单击			

脚本组件1.清除 ()

脚本组件1.语言 = 语言框.取项目文本 (语言框.现行选中项)

如果真 (脚本组件1.执行 (代码框.内容) = 假)

错误框.内容 = 脚本组件1.错误信息

具体脚本代码的含义在此不做具体讲解，请读者自行查阅相关资料。

**注意：**使用脚本程序直接对外部可执行文件进行操作时，部分杀毒软件会认为是病毒代码而报警或自动清除。建议在编写脚本文件时尽量使用标准对象和函数对系统进行操作，以避免不必要的麻烦。

用脚本语言 S.E.S



## 第二十七章 Word 2000 支持库

易语言 3.7 版本推出后,新增加了 Word 2000 支持库,可以直接控制 Word 2000 以上版本。本章将介绍如何使用 Word 2000 支持库。

### 27.1 Word 2000 支持库简介

Word 2000 支持库由 3 个组件组成,包括“Word 程序”组件、“Word 文档集”组件和“Word 图形”组件。

“Word 程序”组件用来创建一个 Word 主程序,并对 Word 主程序进行控制。

“Word 文档集”组件用来创建新 Word 文档和打开 Word 文档等,并对 Word 文档进行操作。在创建 Word 文档前要先创建 Word 的主程序。

“Word 图形”组件用来向文档中添加“图形”、“图片”和“艺术字”等。在向文档中添加图片前要先创建一个 Word 文档或打开一个 Word 文档。

以上 3 个组件通常都是互相配合来使用的,使用时要将几个组件连接起来。

### 27.2 Word 2000 支持库组件

#### 27.2.1 “Word 程序”组件

##### 1. “Word 程序”组件的重要属性

“Word 程序”组件除公有属性外,其他都是“设计时不可用”的属性,这些属性只有在创建 Word 后,才能进行读取或赋值。

##### (1) “左”、“顶”、“宽”、“高”属性

这 4 个属性控制 Word 程序的“左边”、“顶边”、“宽度”和“高度”,在 Word 创建后可以对这 4 个属性进行赋值来控制 Word 窗口在屏幕中的位置及大小。

##### (2) “窗口标题”和“窗口状态”属性

“窗口标题”属性是 Word 程序的窗口标题,Word 创建后,可以更改当前 Word 的窗口标题。





“窗口状态”属性，控制 Word 程序窗口的“正常”、“最大化”和“最小化”，该属性是整数型的属性，0 代表正常，1 代表最大化，2 代表最小化。

### (3) “显示”属性

该属性相当于组件的“可视”属性，可以控制创建的 Word 是否可视。当 Word 创建后默认的都是不可视的，所以只有将“可视”属性设置为“真”，才能看到创建出来的 Word 程序。

### (4) “全屏显示”和“显示比例”属性

“全屏显示”属性控制创建后的 Word 程序是否全屏显示，为逻辑型属性。

“显示比例”属性控制 Word 显示文字的比例，和 Word 工具条中的“显示比例”作用是相同的。

### (5) “是否已创建”属性

该属性为只读属性，检查“Word 程序”组件是否已经创建了 Word 程序。

## 2. “Word 程序”组件的重要方法

### (1) “创建 ()”方法

该方法用来创建 Word 程序实例。要想对 Word 程序进行操作，首先要创建 Word 程序实例。

例如：

Word 程序 1. 创建 ()

Word 程序 1. 显示 = 真

以上例程为：用 Word 程序组件创建一个 Word 程序并让其显示出来。

### (2) “清除 ()”方法

将创建的 Word 程序释放掉，要想再次控制 Word 必须重新使用“创建 ()”方法创建。“清除 ()”方法不能将 Word 程序关闭，只是解除 Word 程序与“Word 程序”组件之间的联系。

### (3) “取程序对象 ()”方法

将创建后的 Word 程序对象取出并返回，返回值的类型为“对象”，可以使用一个“对象”类型的变量存放该返回值，然后使用操作对象的方法来调用该对象的属性、事件及方法。关于对象的操作可以参见“第十七章 COM 组件”中的介绍。

例如：

变量名	类型	静态	数组	备注
WORD程序变量	对象			

Word程序1. 创建 ()

WORD程序变量 = Word程序1. 取程序对象 ()

WORD程序变量. 查看 ()

上述程序在创建一个 Word 程序后，将该对象存放在“对象”变量中，并使用对象的



“查看 ( )”方法查看。

#### (4) “退出 ( )”方法

将创建的 Word 程序结束运行。在执行了“退出 ( )”方法后，程序会自动将该对象释放掉。例如：

Word 程序 1. 退出 ( )

#### (5) “激活窗口 ( )”方法

用来激活 Word 程序的窗口，激活状态即窗口标题变为蓝色，并将窗口切换到顶层。

### 3. “Word 程序”组件的自有事件

#### (1) “文档即将关闭”事件

事件的产生时机：当被打开的 Word 文档将被关闭时触发该事件

#### (2) “文档即将打印”事件

事件的产生时机：当被打开的 Word 文档将被打印时触发该事。

#### (3) “文档即将保存”事件

事件的产生时机：当被打开的 Word 文档将被保存时触发该事件。

#### (4) “文档打开”事件

事件的产生时机：当打开一个 Word 文档时触发该事件

#### (5) “新文档”事件

事件的产生时机：新建一个 Word 文档时触发该事件

#### (6) “退出”事件

事件的产生时机：当 Word 程序将被关闭时，触发该事件。

## 27.2.2 “Word 文档集”组件

### 1. “Word 文档集”组件的重要属性

#### (1) “文档数量”属性

该属性为当前 Word 文档集打开（包括新建）的 Word 文档数量。

#### (2) “内容”属性

内容属性为当前文档中被选中区域的文本内容。如果当前文档中没有被选中区域，“内容”属性为空文本。

#### (3) “字体加粗”、“字体颜色”、“倾斜”、“字体名”、“下划线”、“字体大小”、“删除线”属性

这些属性都和当前选中文字的字体有关，对应“字体通用对话框”中可以设置的项目。例如：





子程序名	返回值类型	公开	备注
改变word及编辑框字体_被选择			

```
--- 如果真 (字体通用对话框. 打开 ())  
    编辑框1. 文本颜色 = 字体通用对话框. 字体颜色  
    编辑框1. 字体. 加粗 = 字体通用对话框. 加粗  
    编辑框1. 字体. 倾斜 = 字体通用对话框. 倾斜  
    编辑框1. 字体. 删除线 = 字体通用对话框. 删除线  
    编辑框1. 字体. 下划线 = 字体通用对话框. 下划线  
    编辑框1. 字体. 字体名称 = 字体通用对话框. 字体名称  
    编辑框1. 字体. 字体大小 = 字体通用对话框. 字体大小  
    Word文档集1. 字体颜色 = 字体通用对话框. 字体颜色  
    Word文档集1. 字体加粗 = 选择 (字体通用对话框. 加粗, 1, 0)  
    Word文档集1. 倾斜 = 选择 (字体通用对话框. 倾斜, 1, 0)  
    Word文档集1. 删除线 = 选择 (字体通用对话框. 删除线, 1, 0)  
    Word文档集1. 下划线 = 选择 (字体通用对话框. 下划线, 1, 0)  
    Word文档集1. 字体名 = 字体通用对话框. 字体名称  
    Word文档集1. 字体大小 = 字体通用对话框. 字体大小
```

程序中当“改变字体”菜单被选择，会弹出字体通用对话框，然后使用字体通用对话框中所设置字体的一系列属性，来改变编辑框中的字体和 Word 文档中当前选中文本的字体。Word 文档集中“字体加粗”、“倾斜”、“删除线”和“下划线”属性的数据类型，和通用对话框的对应属性不同，Word 程序集中的这些属性都是整数型的，所以程序中通过“选择（）”命令将通用对话框的逻辑型属性转换为对应的数值。

(4) “双删除线”、“着重号”“阳文”、“阴文”、“字体隐藏”、“东亚字体名”、“空心”“文字位置”、“字体缩放比”、“阴影”、“删除线”、“上标”、“下标”、“大写格式”、“小写字母”属性

Word 文档集的这些属性，都是字体通用对话框中不能设置的，这些属性都可以在 Word 内嵌的“字体对话框”中进行设置，来查看效果。然后可以在程序中对这些属性赋值。这些属性针对的仍是当前被选中文本。例如：

```
Word 文档集 1. 双删除线 = 1  
Word 文档集 1. 着重号 = 1  
Word 文档集 1. 阳文 = 1  
Word 文档集 1. 空心 = 1  
Word 文档集 1. 阴影 = 1  
Word 文档集 1. 上标 = 1
```

以上代码设置 Word 文档集 1 中当前文档中被选中文本的对应字体效果。给这几个属性赋值 1，表示真，即设置该效果；如果为 0 则表示假，即取消效果。

“Word 文档集”组件的属性还有很多，这里就不一一说明，可以查看易语言的帮助文档。



## 2. “Word 文档集”组件的重要方法

下面对“Word 文档集”组件的几个常用方法进行介绍。

### (1) “置程序 ()”方法

本方法有一个类型为“Word 程序”的参数。

将本 Word 程序集与参数所指定的 Word 程序关联起来，成功返回真，失败返回假。注意：在使用“Word 文档集”的任何方法及属性前必须先使用本方法，设置 Word 文档集组件所属的 Word 程序实例。例如：

子程序名	返回值类型	公开	备注
_新建word文档_被选择			

变量名	类型	静态	数组	备注
文档对象	对象			

```
--- 如果 (Word程序1.是否已创建 = 假)
```

```
Word程序1.创建 ()
```

```
Word程序1.显示 = 真
```

```
▶ 文档对象 = Word文档集1.取文档对象 ()
```

```
文档对象.通用方法 (“close”, )
```

```
Word文档集1.置程序 (Word程序1, 真)
```

```
要保存的文档名 = “”
```

```
_启动窗口.标题 = “word版本为：” + Word程序1.版本
```

上述子程序中，首先判断 Word 程序是否创建，如果没有创建则创建一个 Word 程序。如果 Word 程序已经创建，则使用了“取文档对象 ()”的方法将 Word 文档集对象取出并存放在“文档对象”变量中。由于希望一个“Word 文档集”只操作 1 个 Word 文档，所以在新建一个 Word 文档前，先将前一个 Word 文档关闭。

然后使用“Word 文档集”组件的“置程序 ()”方法，将 Word 文档集所关联的 Word 程序对象设置为“Word 程序 1”。“置程序 ()”方法的第二个参数设置为“真”，表示在置程序的同时新建一个空文档。

也可以将“置程序 ()”方法的第二个参数设置为“假”，然后使用“添加文档 ()”方法添加新文档。

子程序的最后，将“要保存的文档名”文本变量清空，让启动窗口的窗口标题显示 Word 的版本号。

### (2) “打开 ()”方法

该方法用来打开指定的 Word 文档，其参数是要打开的 Word 文档的含完整路径文件名。例如：





子程序名	返回值类型	公开	备注
_打开word文档_被选择			

变量名	类型	静态	数组	备注
文档对象	对象			

```
如果真 (打开通用对话框. 打开 ())
    如果 (Word程序1. 是否已创建 = 假)
        Word程序1. 创建 ()
        Word程序1. 显示 = 真
    Word文档集1. 置程序 (Word程序1, 假)
    文档对象 = Word文档集1. 取文档对象 ()
    文档对象. 通用方法 ("close", )
    Word文档集1. 打开 (打开通用对话框. 文件名)
    要保存的文档名 = 打开通用对话框. 文件名
```

程序中，首先弹出“打开通用对话框”，来选择欲打开的 Word 文档，然后判断 Word 程序是否已创建。如果没创建，则创建一个 Word 程序，并使用 Word 文档集的“置程序()”方法将文档集与 Word 程序关联；如果当前 Word 程序已经创建，则使用 Word 文档对象的“close”方法来关闭当前的 Word 文档。最后使用“打开()”方法，打开通用对话框中选中的文件，并将打开的文件名存放在“要保存的文档名”变量中。

### (3) “保存()”方法

用一个新的文件名及默认格式来保存指定文档。例如：

子程序名	返回值类型	公开	备注
_保存word文档_被选择			

```
如果 (要保存的文档名 ≠ "")
    Word文档集1. 保存 (要保存的文档名)
    如果真 (保存通用对话框. 打开 ())
        Word文档集1. 保存 (保存通用对话框. 文件名)
```

上述程序中，如果“要保存的文档名”变量不为空，即打开过一个 Word 文档，则使用“保存()”方法保存到已打开的文档中。如果没有打开过 Word 文档，则弹出“保存通用对话框”来输入欲保存的文件名。

选择例程中的“程序”菜单的“Word 文档另存为”选项，在“\_ Word 文档另存为\_被选择”子程序中输入代码：

子程序名	返回值类型	公开	备注
_word文档另存为_被选择			

```
如果真 (保存通用对话框. 打开 ())
    Word文档集1. 保存 (保存通用对话框. 文件名)
    要保存的文档名 = 打开通用对话框. 文件名
```

当选择了“Word 文档另存为”则弹出“保存通用对话框”，来输入欲另存为的 Word



文件名，然后将另存为的文件名存放到“要保存的文档名”变量中。

#### (4) “输入文本 ()”方法

插入指定的文本，或者输入键盘命令。该方法的格式为：

Word 文档集. 输入文本 (输入类型, 输入内容)

“输入类型”参数为整数型，初始值为“0”。可以是以下常量值之一：0、输入字符(后面的文本参数起作用)；1、输入回车；2、输入 Back Space。例如：

子程序名	返回值类型	公开	备注
_输入文本按钮_被单击			

Word 文档集1. 输入文本 (0, 编辑框1. 内容)

按钮被按下后，将编辑框中的内容输入到当前的 Word 文档中。

以上代码出自例程：“Word 例程. e”。

### 27.2.3 “Word 图形”组件

Word 图形组件用来对 Word 文档中添加图形、图片及艺术字等等，该组件没有重要的属性及事件，只有组件的方法。

#### 1. “置文档 ()”方法

设置 Word 文档，注意：在使用本对象的任何方法及属性前必须先使用本方法。在方法的参数中填写一个文档集组件名称。例如：

Word 图形. 置文档 (Word 程序集)

#### 2. “清除 ()”方法

将 Word 图形对象释放。再次使用的时候需再次使用“置文档”方法。

#### 4. “添加图片 ()”方法

向文档中的指定位置添加一张图片。

#### 5. “添加图形 ()”方法

向文档中的指定位置添加一个图形。添加的图形为 Word 中选择菜单“插入”→“图片”→“自选图形”功能中添加的自选图形。命令格式为：

Word 图形. 添加图形 (图形类型, 左, 顶, 宽, 高)

“图形类型”参数，整数型，初始值为“1”。指定要创建的自选图形的类型。其值可取任意的 MsoAutoShapeType 常量，可以是-2 以及 1~138 共 139 个图形。

“左”、“顶”、“宽”、“高”参数，控制添加图形的位置及大小。

#### 6. “添加文本框 ()”方法

向文档中的指定位置添加一个文本框。命令格式：

Word 图形. 添加文本框 (内容, 方向, 左, 顶, 宽, 高)

“内容”参数，文本型。是文本框中的文本。

“方向”参数，整数型，初始值为“3”。指定文本的方向。其值可取下列 MsoTextOrientation 常量之一：-2、#混合方向；1、#水平；2、#向上；3、#向下；4、#远东垂直；5、#垂直；6、#远东水平旋转。





“左”、“顶”、“宽”、“高”参数，控制文本框的位置及大小。

### 7. “添加艺术字 方法”

向文档中的指定位置添加一个 Word 艺术字。命令格式为：

Word 图形. 添加艺术字 (内容, PresetTextEffect, 字体名, 字体大小, 粗细, 倾斜, 左, 顶)

“内容”参数，文本型。艺术字的内容。

“PresetTextEffect”参数，整数型，初始值为“0”。预设的文本效果。其值可取 MsoPresetTextEffect 常量，为-2 及 0~29 共 31 个文本效果。

“字体名”参数，文本型，初始值为“宋体”。为艺术字的字体的名称。

“字体大小”参数，小数型，初始值为“36”。为艺术字的字体的大小。

“粗细”和“倾斜”参数，逻辑型。艺术字是否加粗和倾斜。

“左”和“顶”参数，为艺术字的左坐标和顶坐标。

## 27.3 Word 2000 支持库例程

下面提供一个简单的例程“Word 例程 2. e”，来进一步了解这 3 个组件。

首先新建易程序，进行程序界面的设计。如图 27-1 所示。

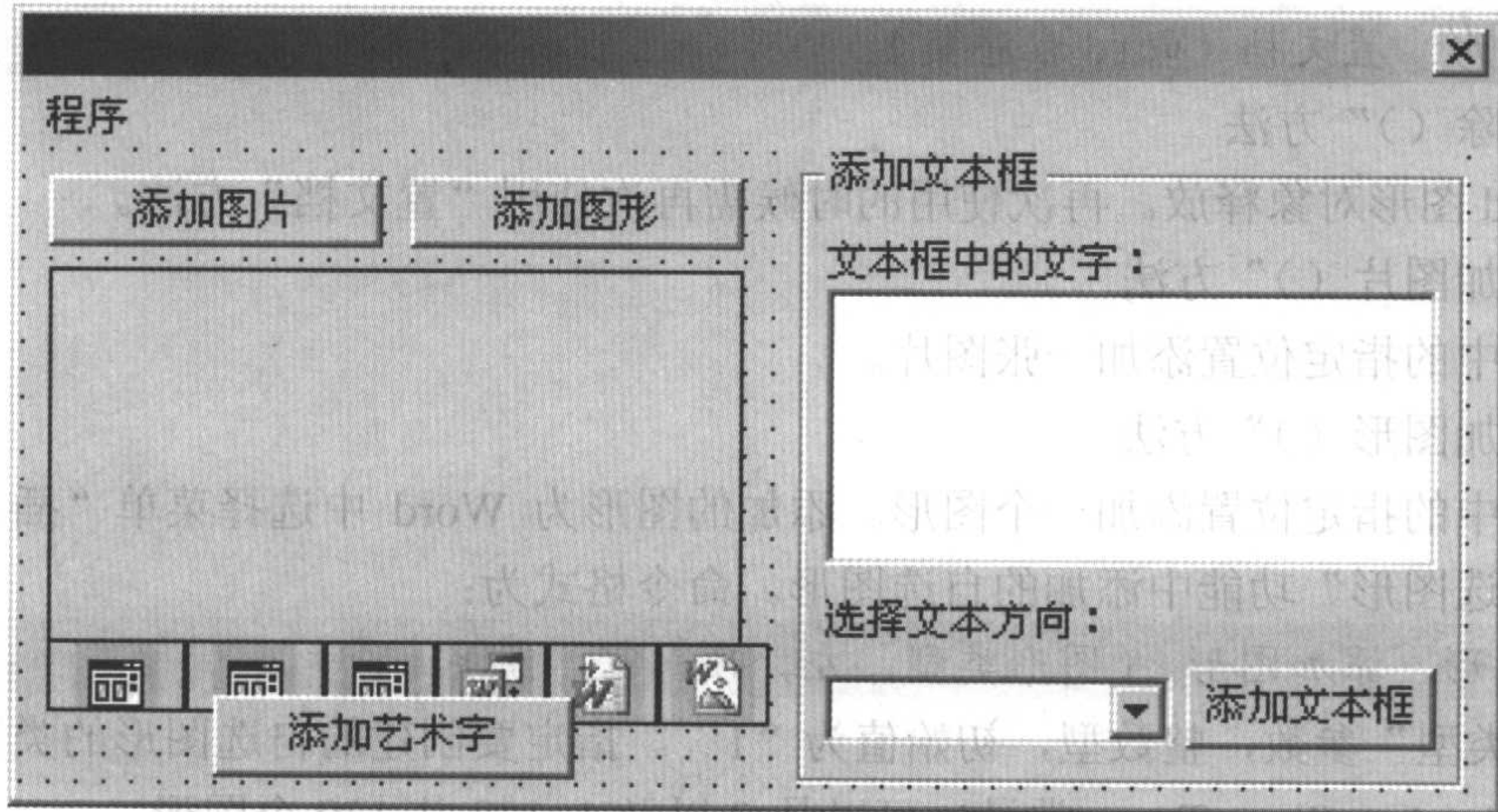


图 27-1 Word 例程界面

窗口中创建 1 个名称为“程序”的菜单，子菜单为：“新建 Word 文档”、“打开 Word 文档”和“保存 Word 文档”。

窗口中添加了 1 个画板组件、分组框组件、编辑框组件和组合框组件。

窗口中还添加了 3 个通用对话框组件及“Word 程序”、“Word 文档集”、“Word 图形”组件。其他的是按钮及标签组件。

例程中，选择“程序”菜单中的“新建 Word 文档”选项，在“\_新建 Word 文档\_被选择”子程序中输入代码：



子程序名	返回值类型	公开	备注
_新建word文档_被选择			

变量名	类型	静态	数组	备注
文档对象	对象			

```

--- 如果 (Word程序1.是否已创建 = 假)
    Word程序1.创建 ()
    Word程序1.显示 = 真
    ▶ 文档对象 = Word文档集1.取文档对象 ()
    文档对象.通用方法 ("close", )
    Word文档集1.置程序 (Word程序1, 真)
    Word图形1.置文档 (Word文档集1)
    要保存的文档名 = ""

```

程序中，首先判断 Word 程序是否已经创建，如果未创建，则创建一个 Word 程序，然后置 Word 文档集程序，并使用 Word 图形组件的“置文档（）”方法，将文档对象设置为“Word 文档集 1”。

双击“添加图片”按钮，在“\_添加图片按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_添加图片按钮_被单击			

变量名	类型	静态	数组	备注
图片号	整数型			

```

--- 如果真 (打开图片通用对话框.打开 ())
    图片号 = 载入图片 (打开图片通用对话框.文件名)
    Word图形1.添加图片 (打开图片通用对话框.文件名, 假, 真, 0, 0, 画板1.取
        图片宽度 (图片号), 画板1.取图片高度 (图片号))
    画板1.底图 = 读入文件 (打开图片通用对话框.文件名)

```

以上程序实现：点击按钮后，弹出“打开图片通用对话框”，选中一个图片后，使用“添加图片（）”方法将该图片添加到已经打开的 Word 文档中，然后将该图片设置为画板的底图。

其中“添加图片（）”方法的宽度和高度参数，是取自己画板组件的图片的实际宽度和高度。

第 2 个窗口名为“添加图形窗口”，在窗口中添加 8 个图片框，然后将几种自选图形的图片添加到图片框中。如图 27-2 所示。



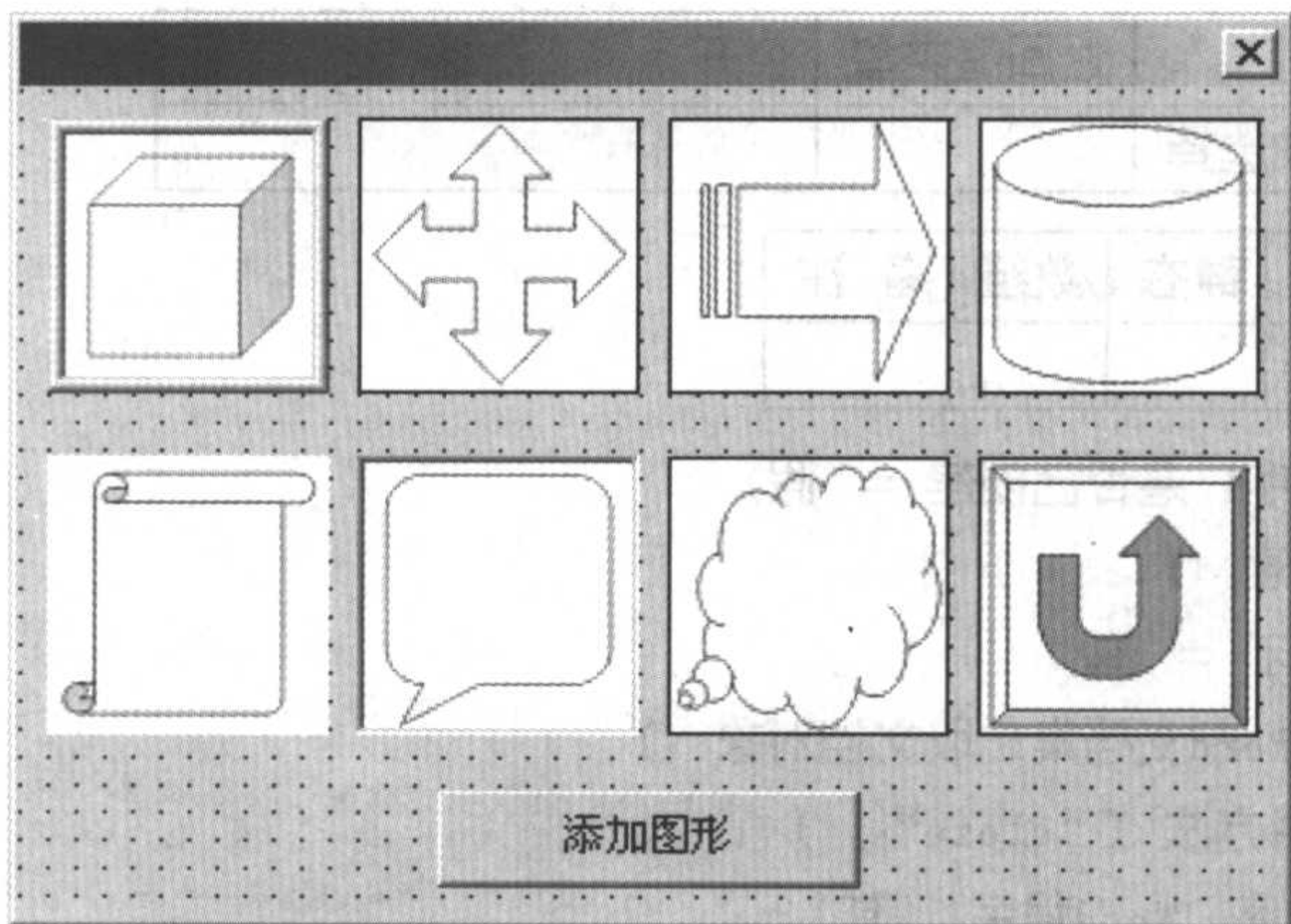


图 27-2 设计界面

新建一个子程序，改子程序名称为：“边框还原”。然后输入以下程序代码，将所有的图片框的边框属性都设置为无边框。程序如下：

子程序名	返回值类型	公开	备注
边框还原			

图片框1.边框 = 0  
图片框2.边框 = 0  
图片框3.边框 = 0  
图片框6.边框 = 0  
图片框7.边框 = 0  
图片框4.边框 = 0  
图片框5.边框 = 0  
图片框8.边框 = 0

然后回到窗口设计界面，双击每一个图片框，进入“鼠标左键被按下”事件子程序中，将当前选中的图片框边框设置为“4 镜框式”边框，这样当选中某一个图片框时，其他边框都无边框，只有当前的图片框显示出边框。程序代码如下：

子程序名	返回值类型	公开	备注		
_图片框1_鼠标左键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

边框还原 0  
图片框1.边框 = 4

再回到窗口设计界面，双击窗口中的“添加图形”按钮，在“\_添加图形按钮\_被单击”



事件子程序中输入代码：

子程序名	返回值类型	公开	备注
_添加图形按钮_被单击			在-添加图形窗口-程序集内

变量名	类型	静态	数组	备注
图形类型	整数型			

```

-- 判断 (图片框1.边框 = 4)
-- 图形类型 = 14
-- 判断 (图片框2.边框 = 4)
-- 图形类型 = 39
-- 判断 (图片框3.边框 = 4)
-- 图形类型 = 49
-- 判断 (图片框4.边框 = 4)
-- 图形类型 = 86
-- 判断 (图片框5.边框 = 4)
-- 图形类型 = 101
-- 判断 (图片框6.边框 = 4)
-- 图形类型 = 106
-- 判断 (图片框7.边框 = 4)
-- 图形类型 = 108
-- 判断 (图片框8.边框 = 4)
-- 图形类型 = 133
-- 信息框 ("请选择一种图形", 0, )
-- 返回 0
_启动窗口.Word图形1.添加图形 (图形类型, 0, 0, 100, 100)

```

上述代码中判断语句的作用是：判断哪一个图片框被选中，若选中就向 WORD 文档中添加那个图形。

最后双击启动窗口中的“添加图形”按钮，在“\_添加图形按钮\_被单击”子程序中输入如下代码：

子程序名	返回值类型	公开	备注
_添加图形按钮_被单击			_启动窗口中的子程序

载入 (添加图形窗口, , 假)

以上程序实现了：当“添加图形”按钮被单击，则弹出“添加图形窗口”。

现在可以试运行本程序，新建一个 Word 文档并添加一个图形，如图 27-2 所示。



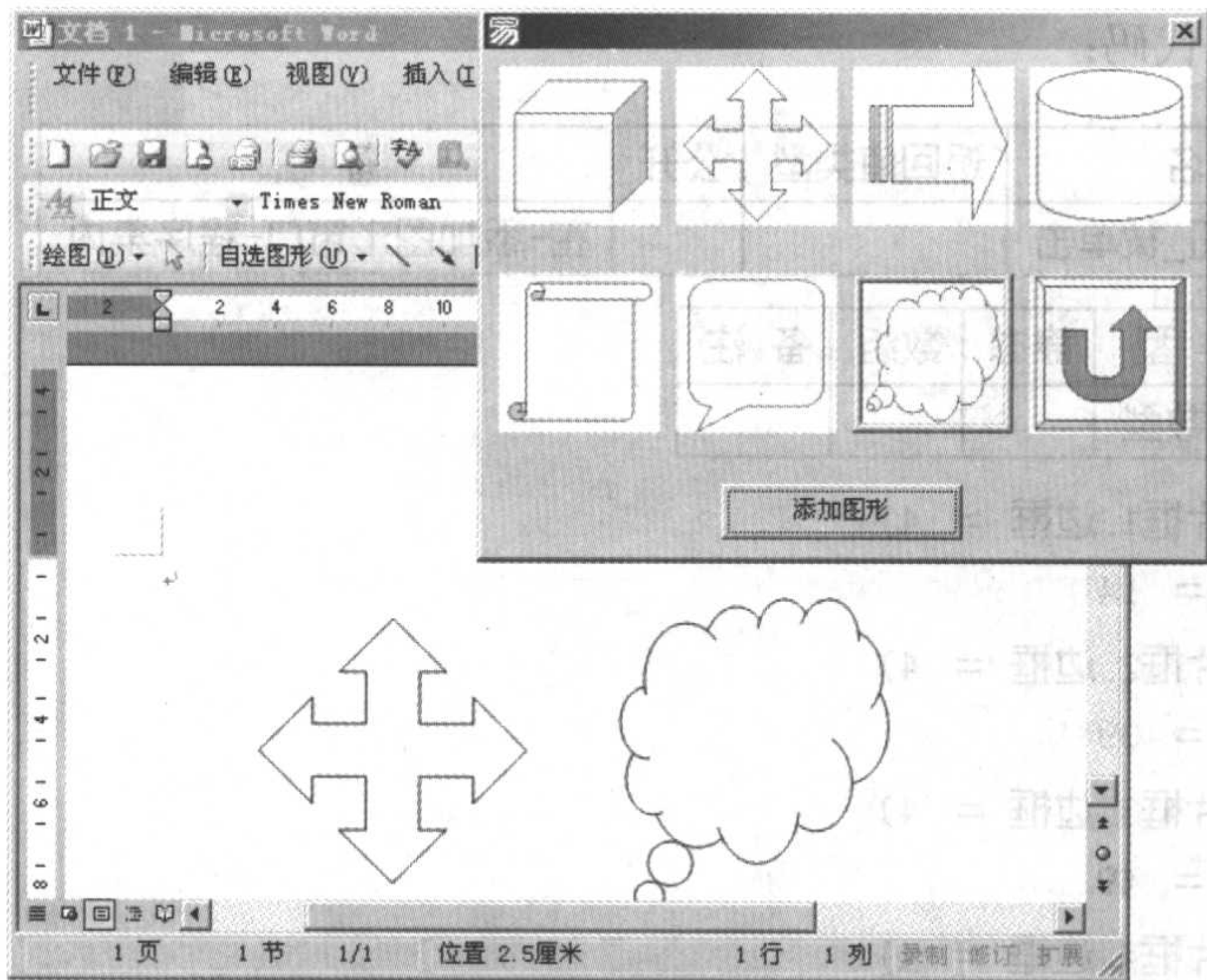


图 27-2 向 Word 中添加图形

回到程序设计界面，将“文本方向组合框”组合框中添加 7 个项目，项目文本为“方向”参数提供的 7 种方向，并将项目对应的常量值添加到组合框的“项目数值”属性中。双击“添加文本框”按钮，在“\_添加文本框按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_添加文本框按钮_被单击			

Word图形1.添加文本框 (编辑框1.内容, 文本方向组合框.取项目数值 (文本方向组合框.现行选中项), 0, 0, 100, 100)

上述程序实现了：添加文本框按钮被单击后，向 Word 文档中添加一个文本框，文本框中文字的方向为组合框中选中的方向，文本框中的文本为编辑框 1 中的内容。

第 3 个窗口名为“添加艺术字窗口”，此窗口中的编辑框用来输入艺术字的内容，6 个图片框中加入了 6 种艺术字样式的图片，并添加了 1 个通用对话框组件。如图 27-3 所示。

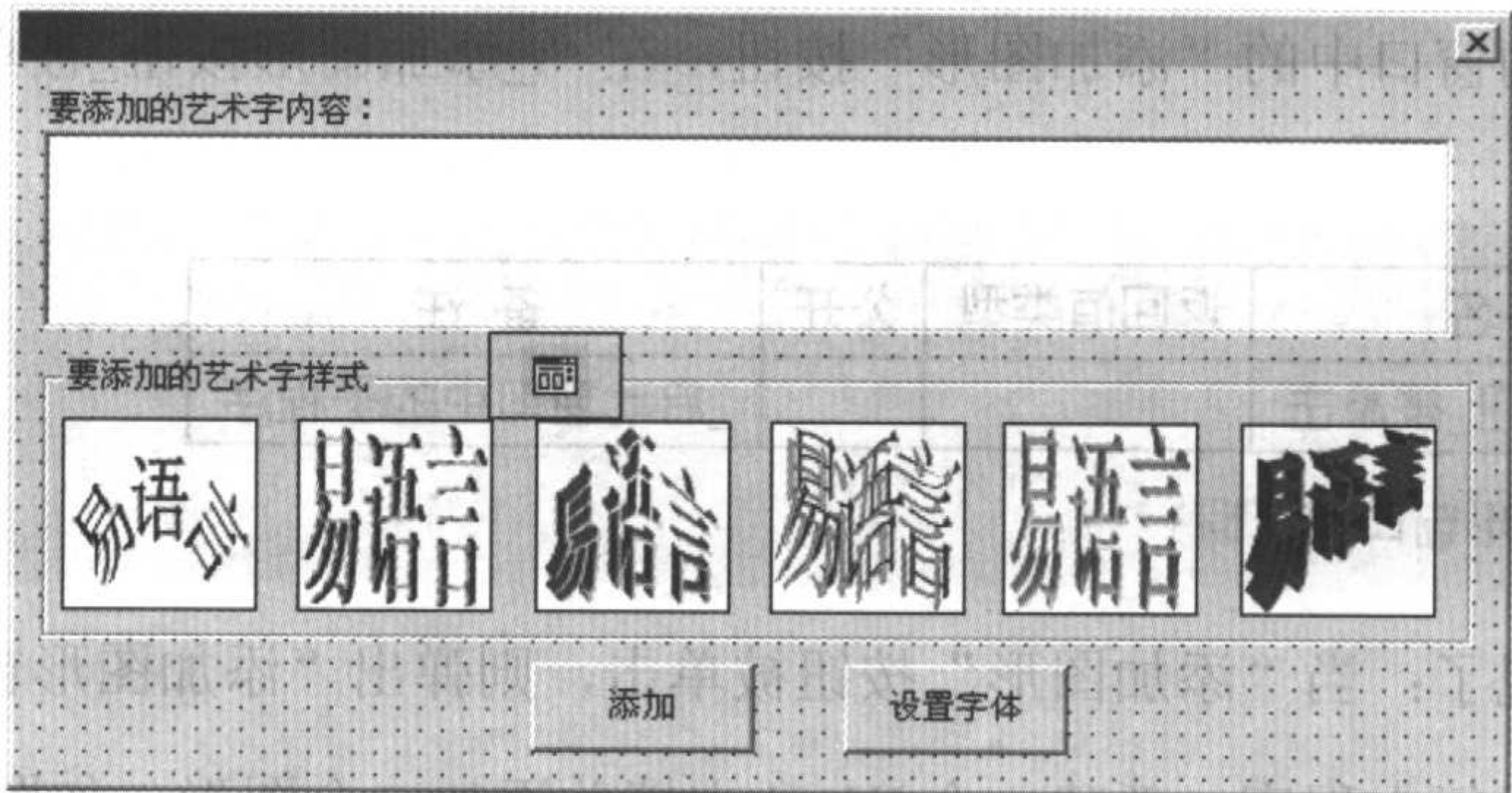


图 27-3 设计程序界面

图片框去边框的“边框还原”子程序与前面的子程序基本一样，在此略。



双击“设置字体”按钮，在“\_设置字体按钮\_被单击”子程序中输入代码：

子程序名	返回值类型	公开	备注
_设置字体按钮_被单击			

--- 如果真 (字体通用对话框. 打开 () )

编辑框1. 字体. 字体名称 = 字体通用对话框. 字体名称

编辑框1. 字体. 字体大小 = 字体通用对话框. 字体大小

编辑框1. 字体. 加粗 = 字体通用对话框. 加粗

编辑框1. 字体. 倾斜 = 字体通用对话框. 倾斜

上述程序实现了使用通用对话框设置编辑框的字体。

双击“添加”按钮，在“\_添加按钮\_被单击”子程序中输入以下代码：

子程序名	返回值类型	公开	备注
_添加按钮_被单击			添加艺术字窗口中.

变量名	类型	静态	数组	备注
添加类型	整数型			

--- 判断 (图片框1. 边框 = 4)

添加类型 = 2

--- 判断 (图片框2. 边框 = 4)

添加类型 = 18

--- 判断 (图片框3. 边框 = 4)

添加类型 = 19

--- 判断 (图片框4. 边框 = 4)

添加类型 = 21

--- 判断 (图片框5. 边框 = 4)

添加类型 = 24

--- 判断 (图片框6. 边框 = 4)

添加类型 = 25

--- 信息框 (“请选择艺术字类型”, 0, )

信息框 (编辑框1. 字体. 字体名称, 0, )

\_启动窗口. Word图形1. 添加艺术字 (编辑框1. 内容, 添加类型, 编辑框1. 字体. 字体名称, 编辑框1. 字体. 字体大小, 编辑框1. 字体. 加粗, 编辑框1. 字体. 倾斜, 0, 0)

上述程序中首先判断哪个图片框被选中，然后将对应艺术字效果的常量值存放在“添加类型”变量中，然后使用“添加艺术字 ()”方法将编辑框中的内容添加到 Word 文档。

本例程还有部分重复代码没有列出，具体可以直接查看随书光盘中的例程“Word 例程 2. e”。





## 第二十八章 Excel 2000 支持库


Office 2000 软件包中提供了 Excel 2000 软件，这是一个用于建立与使用电子报表的实用程序，常用于处理大量的数据信息，特别适用于数据统计。从易语言 3.8 版本开始，增加了 Excel 2000 支持库，提供了对 Excel 电子表格的操作接口，用户无需亲自去操作 Excel 2000，只需设置 Excel 2000 支持库中的相关属性与方法，就可以完成对 Excel 2000 批量生成电子报表等手工操作很烦琐的工作。本章将通过一个例程，向大家介绍如何使用易语言的 Excel 2000 支持库来创建数据表以及根据该数据表中的数据来创建数据型图表。

### 28.1 Excel 2000 支持库简介

在 Excel 2000 支持库中有三个组件，按级别分：

“Excel 程序”组件用来创建一个 Excel 主程序，并对 Excel 主程序进行控制。

“Excel 工作簿”组件用来创建、操作 Excel 工作簿和创建数据表。在创建 Excel 工作簿前要先创建 Excel 的主程序，并把“Excel 工作簿”与“Excel 程序”关联起来。

“Excel 图表”组件用来控制 Excel 工作簿中的图表。在向文档中添加图表前必须先创建一个 Excel 程序和打开一个 Excel 工作簿，并把“Excel 图表”与“Excel 工作簿”关联起来。

### 28.2 Excel 2000 支持库的组件

Excel 支持库中，各组件的属性都比较简单，一些常用的属性将结合最后的例程进行介绍，下面重点介绍组件的重要方法。

#### 28.2.1 “Excel 程序”组件

“Excel 程序”组件的重要方法

##### 1. “创建（）”方法

创建一个“Excel 程序”实例。**注意：**在本支持库的任何操作前，必须先使用本方法来激活组件，以便使用该组件的所有属性与方法。



## 2. “清除 ()” 方法

释放“Excel 程序”组件的操作功能。再次使用的时候需再次使用“创建 ()”方法来创建“Excel 程序”。

## 3. “退出 ()” 方法

退出 Excel 程序。

## 4. “激活窗口 ()” 方法

激活 Excel 程序窗口。

## 5. “模拟按键 ()” 方法

将击键的名称发送给正在运行的 Excel 程序, 让 Excel 程序完成相应的按键事件。例如: 按下“F1”键使 Excel 程序打开相应的帮助文档, 可以通过设置以下程序代码来完成这个按键事件:

```
Excel 程序.模拟按键 (“{f1}”, 假)
```

## 28.2.2 “Excel 工作簿” 组件

### “Excel 工作簿” 组件的重要方法

#### 1. “置程序 ()” 方法

设置“Excel 工作簿”与“Excel 程序”进行关联, 以此来激活组件。注意, 在使用“Excel 工作簿”之前, 必须先使用本方法来激活组件, 以便使用该组件的所有属性与方法。

#### 2. “清除 ()” 方法

释放“Excel 工作簿”的操作功能。再次使用的时候需再次使用“置程序 ()”方法使之与“Excel 程序”进行关联。

#### 3. “合并 ()” 方法

用于合并表格中的单元格。

#### 4. “解除合并 ()” 方法

用于解除表格中合并的单元格。

#### 5. “自动调整 ()” 方法

可以根据所添加内容的多少自动调整单元格大小。

#### 6. “置单元格边框 ()” 方法

设置单元格的边框样式, 其中包括边框的线条样式, 边框粗细和边框颜色。

#### 7. “打开 ()” 方法

打开一个 Excel 2000 的表格文件。

#### 8. “保存 ()” 方法

把指定的 Excel 工作簿保存成“.xls”(Excel 文档)格式。

## 28.2.3 “Excel 图表” 组件

### “Excel 图表” 组件的重要方法





### 1. “置表格 ( )” 方法

设置“Excel 图表”与“Excel 工作簿”进行关联，以此来激活组件。注意，在使用“Excel 图表”之前，必须先使用本方法来激活组件，以便使用该组件的所有属性与方法。

### 2. “清除 ( )” 方法

释放“Excel 图表”的操作功能。再次使用的时候需再次使用“置表格 ( )”方法使之与“Excel 工作簿”进行关联。

### 3. “添加 ( )” 方法

设置好图表的各属性后，用本方法来正式添加图表到表格中。

### 4. “设置系列 ( )” 方法

对图表上显示颜色所代表的数据内容以及图表上数据标签的风格进行设定。其中：“数据标签”参数，用来设置图表上数据标签的风格。值 2，为显示数值；值 3，为显示百分比（仅用于饼图或圆环图）；值 4，为显示标签名称（系列名）；值 5，为显示名称与百分比；值 6，为显示双倍尺寸。默认值为 2。

## 28.3 Excel 2000 支持库例程

这个例程是通过设置“Excel 2000 支持库”的相关属性与方法，对 Excel 2000 的程序进行直接操作，其中包括运行并创建工作簿、创建表、创建图表；下面是设计时的易语言界面。如图 28-1 所示。



图 28-1 例程设计界面

在写程序之前，需要把“Excel 程序”组件、“Excel 工作簿”组件和“Excel 图表”组件添加到“\_启动窗口”中。

运行并创建 Excel 工作簿：

先来实现运行并创建 Excel 程序，并在其中创建一个工作簿。运行“Excel 2000”程序的方法很简单，只需用“Excel 程序”组件的“创建”方法，并让该程序的“显示”属性为真就可以了。通过“模拟按键”方法来模拟键盘上“F1”键的按键事件，打开 Excel 的帮助文档。在“版本号标签”上显示出当前 Excel 程序的版本号和该程序所能使用的打印机名称。最后通过“Excel 工作簿”的“置程序 ( )”方法，设置与“Excel 程序”之间的关联，



以此方法新建一个 Excel 工作簿（注：如果不进行关联将无法对 Excel 工作簿进行设置）。下面就是“\_运行钮\_被单击”事件子程序中的代码：

```

--- 如果 (Excel程序.创建 () = 真)
    Excel程序.显示 = 真
    Excel程序.激活窗口 ()
    Excel程序.模拟按键 (“{f1}”, 假)
    版本号标签.标题 = “Excel程序版本:” + Excel程序.
        版本 + #换行符 + “打印机名称:” + Excel程序
        .打印机
    Excel工作簿.置程序 (Excel程序)
--- _启动窗口.标题 = “Excel程序创建成功”
--> _启动窗口.标题 = “Excel程序创建失败”
    
```

创建表：

接下来，创建一个数据表，表中的数据内容是通过程序随机生成的，如表 28-1 所示。

## 1999 年大富翁总资产透视表

国家	美国	日本	沙特阿拉伯	中国
人数	2	3	2	5
资产最大值 (万元)	931	948	956	916
资产最小值 (万元)	541	631	693	589
资产平均值 (万元)	736	1291	2761	1886
资产总计 (万元)	1472	3873	5522	9430

表 28-1 该例程所建的数据表

数据表包括表的标题和表中的相关内容。在“\_创建表格钮\_被单击”事件中，这两项工作分别调用两个自定义子程序来完成：“置表标题”和“置表内容”。在该事件子程序中，将 Excel 程序的窗口标题设置为“易语言 MSEXcel 操作”，并把工作簿中的英文表格名称改为中文。程序代码如下：

变量名	类型	静态	数组	备注
记次变量	整数型			

```

Excel程序.标题 = “易语言 MSEXcel操作”
--> 计次循环首 (Excel工作簿.表格数量, 记次变量)
    Excel工作簿.表格序号 = 记次变量
    Excel工作簿.表格名 = “表格” + 到文本 (记次变量)
--- 计次循环尾 ()
置表标题 ()
置表内容 ()
    
```





下面看一下“置表标题”子程序的代码：要在“Excel 工作簿”的“表格 1”中创建表，首先要把“表格序号”设置为 1。设置所要显示的区域是从“B1”到“F1”之间；合并这些单元格；设置该单元格的行高和列宽；添加内容为“1999 年大富翁总资产透视表”，并对这些字设置一些相应的字体属性。程序代码如下：

```
Excel工作簿.表格序号 = 1
Excel工作簿.首单元格 = "B1"
Excel工作簿.尾单元格 = "F1"
Excel工作簿.合并 ()
Excel工作簿.行高 = 40
Excel工作簿.列宽 = 15
Excel工作簿.内容 = "1999年大富翁总资产透视表"
Excel工作簿.字体加粗 = 真
Excel工作簿.字体大小 = 30
Excel工作簿.字体阴影 = 真
Excel工作簿.字体名 = "隶书"
Excel工作簿.字体颜色 = 取颜色值 (62, 151, 145)
Excel工作簿.水平对齐方式 = -4108
Excel程序.显示比例 = 85
```

根据上面对数据表标题的创建，大家可以从了解到添加单元格内容的一个简单的过程：

1. 设置表格序号，以确定在当前工作簿的指定表格中进行操作。
2. 设置表的显示区域（包括设置“首单元格”和“尾单元格”）。
3. 如果区域不在同一个格中，需对所选区域进行“合并”。
4. 也可以设置单元格的“行高”和“列宽”。

5. 向所设置好的单元格中添加文字“内容”，并可对这些文字内容的字体属性进行相应的设置。

数据表的创建，就是通过设置每一个单元格的属性，并为每个单元格添加内容的重复过程。

下面看一下“置表内容”子程序的代码：在这个所要创建的表中，包含富翁的“资产最大值”、“资产最小值”、“资产平均值”、“资产总计”和富翁的“人数”。这些数值分别用变量“最大值”、“最小值”、“平均值”、“总计”、“人数”来保存，并且随机生成这些数值。该程序中还有一个计次循环，用于把随机生成的资产值、平均值和资产总计放入表中相应的位置上。“循环变量”就是用于保存该计次循环的次数，文本型变量“格”用来保存表格中表格列所对应的英文字母。下面就是“置表内容”事件子程序的部分相关代码：

```
' 表格第一行
置单元格内容 ("B2", "B2", "国家")
置单元格内容 ("C2", "C2", "美国")
置单元格内容 ("D2", "D2", "日本")
```



```
置单元格内容 ("E2", "E2", "沙特阿拉伯")
置单元格内容 ("F2", "F2", "中国")
' 表格第一列
置单元格内容 ("B3", "B3", "人数")
置单元格内容 ("B4", "B4", "资产最大值(万元)")
置单元格内容 ("B5", "B5", "资产最小值(万元)")
置单元格内容 ("B6", "B6", "资产平均值(万元)")
置单元格内容 ("B7", "B7", "资产总计(万元)")
' 表格第二列
置单元格内容 ("C3", "C3", "2")
置单元格内容 ("D3", "D3", "3")
置单元格内容 ("E3", "E3", "2")
置单元格内容 ("F3", "F3", "5")
' 表格第三至第六列
--变量循环首 (4, 7, 1, 循环变量)
--判断 (循环变量 = 4)
--格 = "C"
--人数 = 2
--随机取值 (人数, 临时数组)
--数组中取值 (人数, 临时数组, 最大值, 最小值, 总计, 平均值)
--判断 (循环变量 = 5)
```

在这个子程序中，还调用了很多自定义子程序，如“随机取值”子程序，随机生成表中富翁的资产值，并对其进行从大到小的排列后保存到“临时数组”中；“数组中取值”子程序，取出临时数组中的“最大值”、“最小值”并对其数组中的所有数值进行“总和”和求出“平均值”。最后，程序中所生成这些数值都将通过“置单元格内容”子程序，添加到相应的表格区域当中。下面就一同来看看这些子程序中的相关代码：

“置单元格内容”子程序的相关代码：

子程序名	返回值类型	公开	备注		
置单元格内容					
参数名	类型	参考	可空	数组	备注
首单元	文本型				
尾单元	文本型				
单元格内容	文本型				

```
Excel工作簿.首单元格 = 首单元
Excel工作簿.尾单元格 = 尾单元
Excel工作簿.内容 = 单元格内容
Excel工作簿.是否适应列宽 = 真
Excel工作簿.水平对齐方式 = -4108
Excel工作簿.置单元格边框 (1, 2, #蓝色)
```





“数组中取值”子程序的相关代码:

参数名	类型	参考	可空	数组	备注
人数	整数型				
数组	整数型			✓	
最大值	整数型	✓			
最小值	整数型	✓			
总和	整数型	✓			
平均值	整数型	✓			

变量名	类型	静态	数组	备注
循环变量	整数型			

最大值 = 数组 [1]

最小值 = 数组 [人数]

--- 计次循环首 (人数, 循环变量)

总和 = 总和 + 数组 [循环变量]

--- 计次循环尾 0

平均值 = 总和 ÷ 人数

“随机取值”子程序的相关代码:

参数名	类型	参考	可空	数组	备注
人数	整数型				
数组	整数型	✓		✓	

变量名	类型	静态	数组	备注
循环变量	整数型			
继续循环	逻辑型			
临时变量	整数型			

重定义数组 (数组, 假, 人数)

--- 计次循环首 (人数, 循环变量)

数组 [循环变量] = 取随机数 (500, 1000)

--- 计次循环尾 0

--- 循环判断首 0 ' 从大到小进行排列

继续循环 = 假

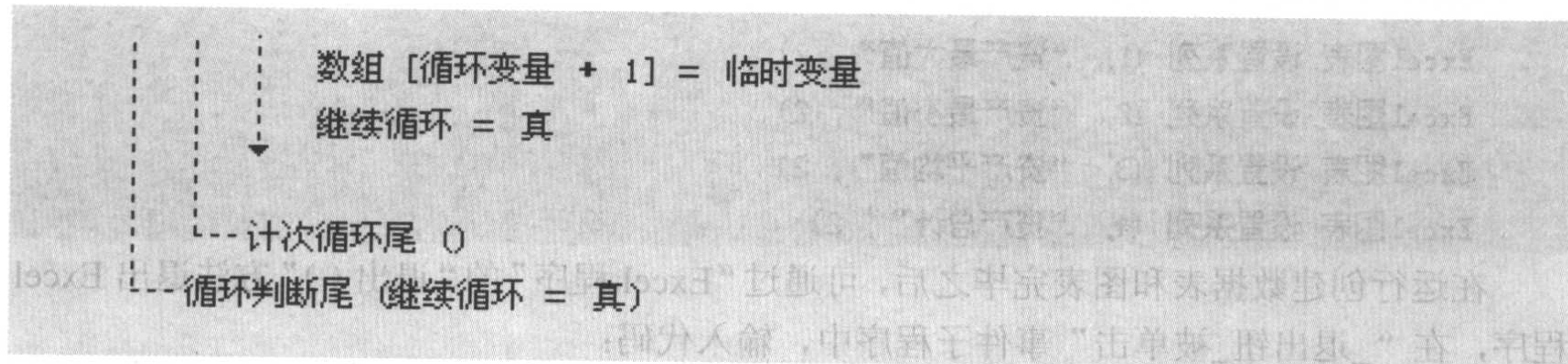
--- 计次循环首 (人数 - 1, 循环变量)

--- 如果真 (数组 [循环变量 + 1] > 数组 [循环变量])

临时变量 = 数组 [循环变量]

数组 [循环变量] = 数组 [循环变量 + 1]





### 创建图表：

先来了解一下创建图表的方法：

1. 置表格，设置“Excel 图表”与“Excel 工作簿”进行关联。
2. 设置图表的标题、图表的左边和顶边的位置，以及图表的宽度和高度；
3. 设置图表中 X 轴和 Y 轴的标题；
4. 设置数据表中所要显示在图表中的内容范围，Y 轴的范围是从数据表中所要显示区域的左上方到所要显示区域的右下方，在易语言中用“Excel 图表”组件的“Y 轴首”和“Y 轴尾”两个属性来设定；X 轴的范围是从数据表中所要显示区域的左边到右边，在易语言中用“Excel 图表”组件的“X 轴首”和“X 轴尾”两个属性来设定。
5. 设置图表的绘制方式和图表的类型。该支持库图表的绘制方式有两种，一种是按行绘制，另一种是按列绘制；在该支持库中图表所能绘制类型有很多，例如“簇状柱形图”、“折线图”、“三维分离型饼图”和“三维堆积柱形图”等等。
6. 用“Excel 图表”的“添加 ()”方法，在 Excel 表格中自动生成图表。

在图表生成完毕后，对图表上显示颜色所代表的数据内容以及图表上数据标签的风格进行设定，只要通过“Excel 图表”组件的“设置系列 ()”方法就可以完成该设定工作。

按照上面所描述的创建图表的思路，来看一下“\_创建图表钮\_被单击”的相关代码：

Excel 图表. 置表格 (Excel 工作簿)

Excel 图表. 标题 = “1999 年大富翁总资产透视表”

Excel 图表. 左 = 75

Excel 图表. 宽 = 420

Excel 图表. 高 = 300

Excel 图表. 顶 = 150

Excel 图表. X 轴标题 = “国家”

Excel 图表. Y 轴标题 = “资产值”

Excel 图表. Y 轴首 = “C2”

Excel 图表. Y 轴尾 = “F6”

Excel 图表. X 轴首 = “C2”

Excel 图表. X 轴尾 = “F2”

Excel 图表. 绘制方式 = 1

Excel 图表. 图表类型 = 4

折线图

Excel 图表. 添加 ()





Excel 图表. 设置系列 (1, “资产最大值”, 2)

Excel 图表. 设置系列 (2, “资产最小值”, 2)

Excel 图表. 设置系列 (3, “资产平均值”, 2)

Excel 图表. 设置系列 (4, “资产总计”, 2)

在运行创建数据表 and 图表完毕之后, 可通过“Excel 程序”的“退出( )”方法退出 Excel 程序, 在“\_退出钮\_被单击”事件子程序中, 输入代码:

Excel 程序. 退出 ( )

运行该程序来看一下运行后的效果, 如图 28-2 所示。

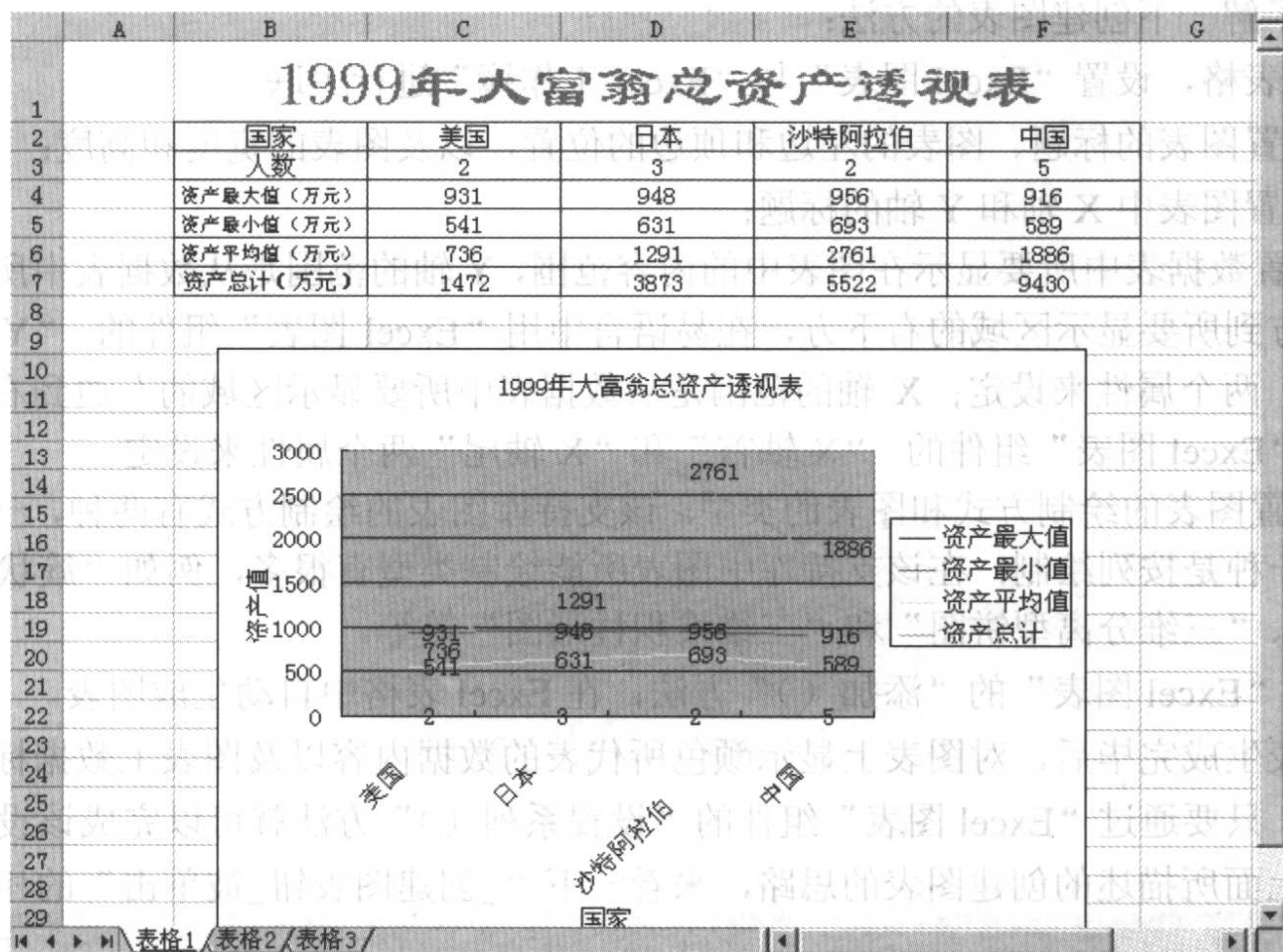


图 28-2 例程运行效果

现在已经学习了两篇 OFFICE 支持库, 现在小结一下。

在 Excel 2000 支持库中包含了三个组件, 按级分为: “Excel 程序” 组件、“Excel 工作簿” 组件、“Excel 图表” 组件, 分别用来运行操作 Excel 程序、创建工作簿和数据表、创建图表。在使用“Excel 工作簿” 组件和“Excel 图表” 组件之前, 必须建立与上一级组件的关联关系, 使组件激活, 以便可以设置该组件的所有属性和方法。关联的方法为:

使用“Excel 工作簿” 组件时: 用“置程序 (“Excel 程序” 组件名称)” 与“Excel 程序” 组件进行关联。

使用“Excel 图表” 组件时: 用“置表格 (“Excel 工作簿” 组件名称)” 与“Excel 工作簿” 组件进行关联。

在创建数据表和图表时, 要注意程序代码的流程步骤, 在前面章节中都有详细的讲解, 在这里就不再赘述了。



## 第二十九章 PowerPoint 2000 支持库

本支持库用来操作 MS PowerPoint 2000/XP/2003 或以上版本，机器中必须装有 PowerPoint 2000 或以上版本。该支持库需要易语言 3.7 版本或更高版本支持。包括的三个功能窗口组件是：PPT 程序、PPT 文稿和 PPT 播放。只有这三个窗口组件配合使用才可以用易语言调用 PowerPoint 2000 实现它的一些基本功能。

### 29.1 PowerPoint 2000 支持库简介

PowerPoint 2000 支持库由 3 个组件组成，包括“PPT 程序”组件、“PPT 文稿”组件和“PPT 播放”组件。

“PPT 程序”组件用来创建一个 PowerPoint 主程序，并对 PowerPoint 主程序进行控制。

“PPT 文稿”组件用来创建新 PPT 文稿和打开旧的 PPT 文稿等，并对 PPT 文稿进行操作。在创建 PPT 文稿前要先创建 PPT 主程序。

“PPT 播放”组件用来控制 PPT 幻灯片的放映。在使用“PPT 播放”组件进行播放前，要先创建一个 PPT 文稿或打开一个 PPT 文稿。

从上面的介绍可以看出，本支持库的这 3 个组件是配合使用的，创建顺序是“PPT 程序”→“PPT 文稿”→“PPT 播放”。

### 29.2 PowerPoint 2000 支持库的组件

#### 29.2.1 “PPT 程序”组件

“PPT 程序”组件的属性比较简单，所以下面重点介绍“PPT 程序”的方法。

##### 1. “创建（）”方法

该方法用来创建一个 PowerPoint 主程序，在本支持库进行任何操作前，必须要先使用本方法创建。

##### 2. “清除（）”和“退出（）”方法

“清除（）”方法，是将易语言在内存中创建的 PowerPoint 对象释放掉，但并不会关



闭 PowerPoint 的主程序，只相当于解除了 PowerPoint 与易语言之间的关联。要再次对 PowerPoint 程序操作时，需再次使用“创建（）”方法创建。

“退出（）”方法，是将创建后的 PowerPoint 主程序关闭，关闭后 PowerPoint 对象也会同时被释放掉。

### 29.2.2 “PPT 文稿” 组件

“PPT 文稿” 组件的重要属性

1. “幻灯片数量” 及 “幻灯片索引” 属性

“幻灯片数量” 属性，反映了当前已经新建或打开的 PPT 文稿中的幻灯片数量。

“幻灯片索引” 属性，表示当前指向的幻灯片的索引，即当前处于被编辑状态的幻灯片的索引。该索引从 1 开始，1 表示第一张幻灯片，2 表示第二张，依此类推。

注意：“幻灯片索引” 是程序内部处于被编辑状态幻灯片索引，并不表示程序当前显示的幻灯片，所以改变幻灯片索引后，程序表面看不出改变，但当对 PPT 文稿进行添加图片、添加图形等操作时，就会以当前幻灯片索引为准。

打开随书光盘本章目录中的“幻灯片设计.e” 例程。程序中“插入幻灯片” 按钮被单击的代码如下：

子程序名	返回值类型	公开	备注
插入幻灯片按钮_被单击			

变量名	类型	静态	数组	备注
幻灯数量	整数型			

```

PPT文稿1.插入幻灯片 (-1, 12)
幻灯数量 = PPT文稿1.幻灯片数量
PPT文稿1.幻灯片索引 = 幻灯数量
删除按钮.禁止 = 假
保存按钮.禁止 = 假
选择夹1.禁止 = 假
选择夹1.现行子夹 = 0
插入组合框.现行选中项 = 0
                    
```

程序中，当插入一张新的幻灯片，则改变“幻灯数量”变量的值为“幻灯片数量”属性的值，然后将当前幻灯片索引设置为新加入的幻灯片。然后改变其他几个组件的属性以配合程序。

2. “切换效果”、“单击切换”、“定时切换”和“切换间隔”属性

“切换效果”属性，如果大家使用或观看过 PowerPoint 的文稿，一定对幻灯片之间多变的切换效果印象很深。该属性，就用来控制切换到当前所编辑的幻灯片的效果。PowerPoint 提供了 80 种切换效果供选择，程序中欲改变“切换效果”属性，需要给该属性赋值对应的常量值，这些常量值可以在易语言的即时帮助中查找到。



如程序中要设定切换效果为“右侧覆盖”，在帮助中查找到的常量名为“#ppEffectCoverRight”，该常量对应的常量值为“1283”，所以改变切换效果是使用代码：

PPT 文稿 1. 切换效果 = 1283

“单击切换”属性，该属性用来设置幻灯片是否被单击时换片。

“定时切换”属性，该属性用来指定幻灯片是否在规定时间内自动切换。

“切换间隔”属性，当设置了“定时切换”后，该属性用来控制自动切换的时间间隔，以秒计算。

在“幻灯片设计.e”例程中，标题为“应用设置”按钮被单击的代码如下：

子程序名	返回值类型	公开	备注
_应用按钮_被单击			

如果 (单击切换选择框.选中 = 真)

PPT文稿1.单击切换 = 真

▶ PPT文稿1.单击切换 = 假

如果 (间隔选择框.选中 = 真)

PPT文稿1.定时切换 = 真

PPT文稿1.切换间隔 = 到数值 (间隔编辑框.内容)

▶ PPT文稿1.定时切换 = 假

当“应用设置”按钮被单击，通过判断“单击切换选择框”是否被选中来设置“单击切换”属性，通过判断“间隔选择框”是否被选中，来设置“定时切换”属性，如果设置“定时切换”为真，则设置切换时间为“间隔编辑框”中输入的切换时间。

PPT 文稿还有很多属性，可以通过易语言帮助中的介绍来使用，例如：“切换隐藏”、“切换速度”、“切换音效”属性，从属性名即可以看出该属性的作用，通过查看帮助中对属性的参数等的介绍，就可以在需要时对这些属性进行设置了。

### “PPT 文稿”组件的重要方法

#### 1. “置程序 ()”方法

该方法用来设置“PPT 文稿”组件所关联的“PPT 程序”，在使用“PPT 文稿”组件的任何方法及属性前必须先使用本方法。

“幻灯片设计.e”例程中，当\_启动窗口创建完毕，则创建 PPT 程序，并使用“PPT 文稿”组件的“置程序 ()”方法，将“PPT 程序”组件与“PPT 文稿”组件相关联，代码如下：





子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

选择夹1.现行子夹 = 0

插入组合框.现行选中项 = 0

PPT程序1.创建 ()

PPT文稿1.置程序 (PPT程序1, 真)

PPT程序1.显示 = 真

## 2. “插入幻灯片 ()” 和 “删除幻灯片 ()” 方法

“插入幻灯片 ()” 方法用来在指定位置中插入新幻灯片, 该方法有 2 个参数, 第一个参数添写要插入幻灯片的索引值, 如果索引值填写-1, 则将幻灯片插入到最后的位置; 第二个为新插入幻灯片的样式, 使用枚举常量“幻灯版式”。

“删除幻灯片 ()” 方法用来删除指定的幻灯片, 在参数中添入要删除的幻灯片索引。

例如, 在 PPT 文稿的最后, 插入一张新幻灯片, 并将版式设置为 Text 版式, 使用代码:

PPT 文稿 1. 插入幻灯片 (-1, #幻灯版式.Text 版式)

## 3. “添加图片 ()” 方法

该方法用来向当前幻灯片中添加图片。

例如“幻灯片设计.e”例程中, “\_图片按钮\_被单击”子程序的代码如下:

子程序名	返回值类型	公开	备注
_图片按钮_被单击			

通用对话框1.过滤器 = “所有图片文件|\*.emf;\*.wmf;\*.jpg;\*.jpeg;\*.jfif;\*.jpe;\*.png;\*.bmp;\*.dib;\*.rle;\*.bmz;\*.gif;\*.gfa;\*.emz;\*.wmz;\*.pcx;\*.tif;\*.tiff;\*.cdr;\*.cgm;\*.eps;\*.pct;\*.pict;\*.wpg”

通用对话框1.类型 = 0

--- 如果真 (通用对话框1.打开 () = 真)

图片编辑框.内容 = 通用对话框1.文件名

PPT文稿1.添加图片 (图片编辑框.内容, 假, 真, 到数值 (媒体左边编辑框.内容), 到数值 (媒体顶边编辑框.内容), 到数值 (媒体宽度编辑框.内容), 到数值 (媒体高度编辑框.内容))

选择夹1.现行子夹 = 1

插入组合框.现行选中项 = 1

程序中, 首先设置了“通用对话框 1”的“过滤器”和“类型属性”, 然后使用通用对话框打开一张图片, 并将该图片添加到当前的幻灯片中。

## 4. “添加背景图片 ()” 方法

该方法用来向当前幻灯片中添加背景图片。

例如“幻灯片设计.e”例程中, “\_背景按钮\_被单击”子程序的代码如下:



子程序名	返回值类型	公开	备注
_背景按钮_被单击			

```
通用对话框1.过滤器 = "所有图片文件|*.emf;*.wmf;*.jpg;*.jpeg;*.jfif;*.jpe;*.png;*.bmp;*.dib;*.rle;*.bmz;*.gif;*.gfa;*.emz;*.wmz;*.pcz;*.tif;*.tiff;*.cdr;*.cgm;*.eps;*.pct;*.pict;*.wpg"
```

```
通用对话框1.类型 = 0
```

```
--- 如果真 (通用对话框1.打开 () = 真)
```

```
背景图编辑框.内容 = 通用对话框1.文件名
```

```
PPT文稿1.添加背景图 (通用对话框1.文件名)
```

```
选择夹1.现行子夹 = 2
```

```
插入组合框.现行选中项 = 2
```

程序中，首先设置了“通用对话框1”的“过滤器”和“类型属性”，然后使用通用对话框打开一张图片，并将打开的图片设置为当前幻灯片的背景图片。

“PPT 文稿”组件还有很多方法，例如“添加媒体 ()”、“添加艺术字 ()”等等，很多方法的使用都类似，大家可以根据易语言帮助的介绍来使用这些方法。

### 29.2.3 “PPT 播放” 组件

“PPT 播放”组件的属性比较简单，大家可以根据易语言帮助的介绍来使用，下面重点介绍几个常用的方法。

#### 1. “置文稿 ()” 方法

该方法用来设置“PPT 播放”组件和“PPT 播放”组件的关联，在使用本“PPT 播放”组件的任何方法及属性前必须先使用本方法。

#### 2. “放映 ()” 方法

控制幻灯片开始放映。

例如，“幻灯片设计.e”例程中“\_放映按钮\_被单击”子程序代码如下：

子程序名	返回值类型	公开	备注
_放映按钮_被单击			

```
PPT播放1.置文稿 (PPT文稿1)
```

```
PPT文稿1.首幻灯片索引 = 1
```

```
PPT播放1.放映 ()
```

当“放映”按钮被单击，则将“PPT 播放”组件与“PPT 文稿”组件关联起来，然后使用了“PPT 播放”组件的“放映 ()”方法，开始播放幻灯片。

#### 3. “结束 ()” 方法

该方法用来结束幻灯片的放映。

#### 4. “切换 ()”、“到首张 ()”、“到尾张 ()”、“下一张 ()”、“上一张 ()” 方法

这些方法都用来控制当前所播放的幻灯片。

“切换 ()”方法可以切换到指定幻灯片。





“到首张 ()”和“到尾张 ()”方法用来切换到第一张和最后一张幻灯片。

“下一张 ()”和“上一张 ()”方法，用来切换到当前幻灯片的上一张和下一张幻灯片。

前面讲的是使用易语言调用 PowerPoint 2000 几个组件的基本使用方法。下面结合例程，实际操作一下 PPT 文稿中幻灯片的播放和切换。

## 29.3 PowerPoint 2000 支持库例程

### 29.3.1 人工切换

打开随书光盘本章目录的“切换播放幻灯片.e”，窗体设计界面如图 29-1 所示。

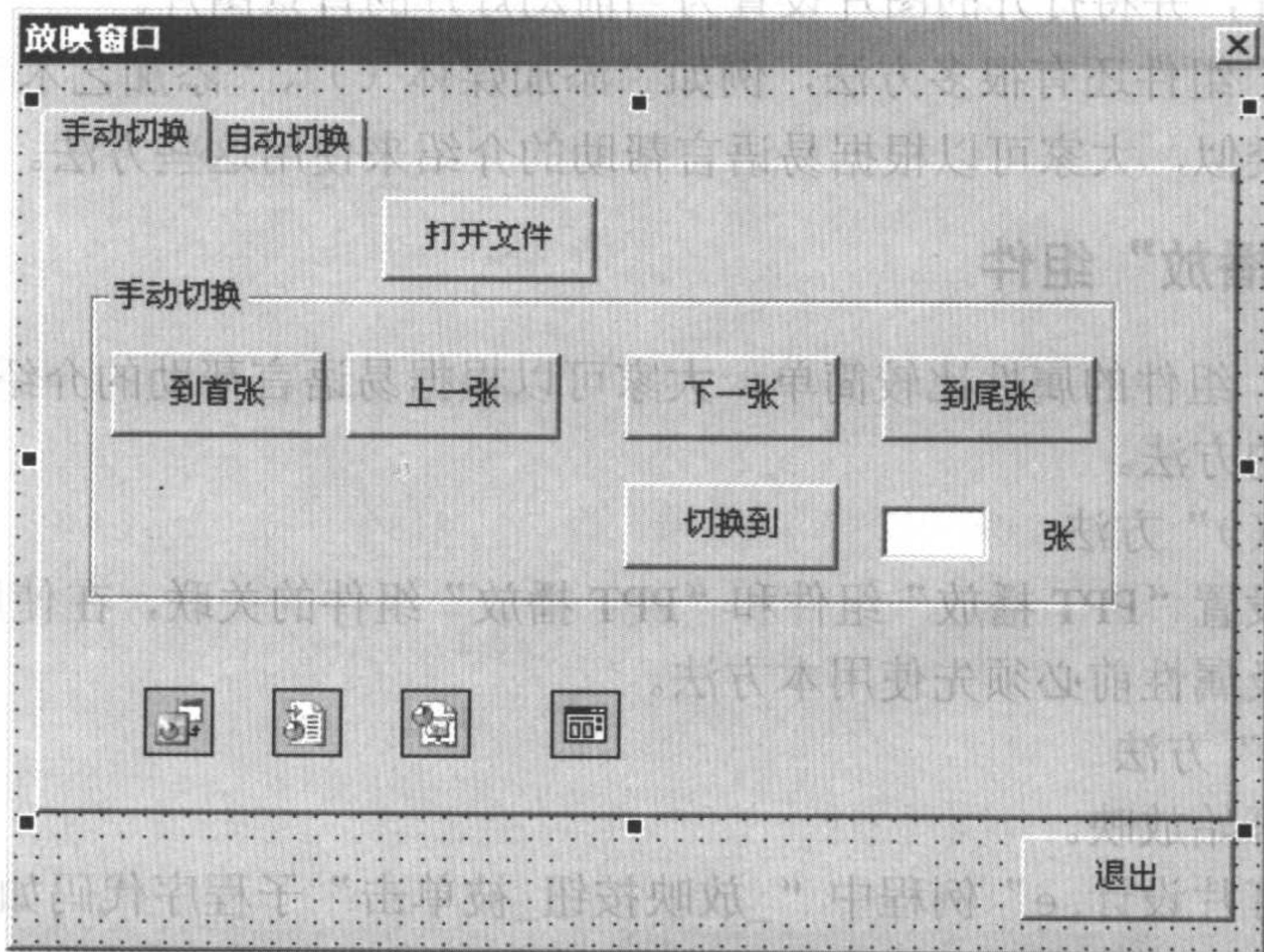


图 29-1 易程序放映窗口

设计窗体上被放置一个选择夹组件，子夹 1 主要使用手动方法控制幻灯片的播放。子夹 2 用来直接按文稿文件的原来设置进行播放或用户自定义设置幻灯片的播放时间间隔。

按 F5 键运行程序，进入放映窗口。首先单击标题为“打开文件”的按钮，通过对话框选择 PPT 演示文稿。相关程序代码如下：

子程序名	返回值类型	公开	备注
_打开文件按钮_被单击			

变量名	类型	静态	数组	备注
局部计次变量	整数型			

```
如果真 (PPT程序1.是否已创建 = 0)  
PPT程序1.创建 ()
```



```

PPT文稿1.置程序 (PPT程序1, 假)
通用对话框1.类型 = 0
通用对话框1.过滤器 = "PPT文稿 (*.ppt)|*.ppt"
--- 如果真 (通用对话框1.打开 () = 真)
    PPT程序1.显示 = 真
    PPT文稿1.打开 (通用对话框1.文件名, 假, 假, 真)
    PPT播放1.置文稿 (PPT文稿1)
    幻灯片数量 = PPT文稿1.幻灯片数量
    幻灯片数量标签.标题 = "幻灯片数量:" + 到文本 (幻灯片数量)
    PPT文稿1.首幻灯片索引 = 1
    PPT播放1.高级模式 = #放映方式.人工
    PPT播放1.放映 ()
    索引变量 = 1
    当前索引标签.标题 = "当前索引:" + 到文本 (索引变量)

```

代码分析如下:

第一个如果真的条件语句,判断当前程序是否已经打开一个 PowerPoint 2000 程序。如果已经创建 (其值为 1),就不再创建新的程序。此程序在第二次打开时有可能无法对幻灯片进行切换,这时就需要退出放映窗口,然后重新载入,再打开文稿文件。

通过对话框选择 PPT 程序,只有将 PPT 程序的显示属性设置为真,使用 PPT 文稿的打开方法才能正确的显示演示文稿。

最后设置 PPT 文稿的首幻灯片索引,同时设置播放的放映方式,然后使用 PPT 播放的放映方法进行放映。

如果打开成功,就可以使用 PPT 播放的“到首张 ()”、“上一张 ()”、“下一张 ()”和“到尾张 ()”等方法进行人工切换。实例代码如下:

PPT 播放.到首张 ()

同时也可以使用 PPT 播放的“切换”方法对幻灯片进行人工切换,在此程序中只使用它切换到指定的幻灯片。实例代码如下:

PPT 播放.切换 (2)

切换到第二张幻灯片。

### 29.3.2 自动播放

上面例程“切换播放幻灯片.e”运行时,点击第二个子夹:“自动切换”子夹界面如图 29-2 所示。



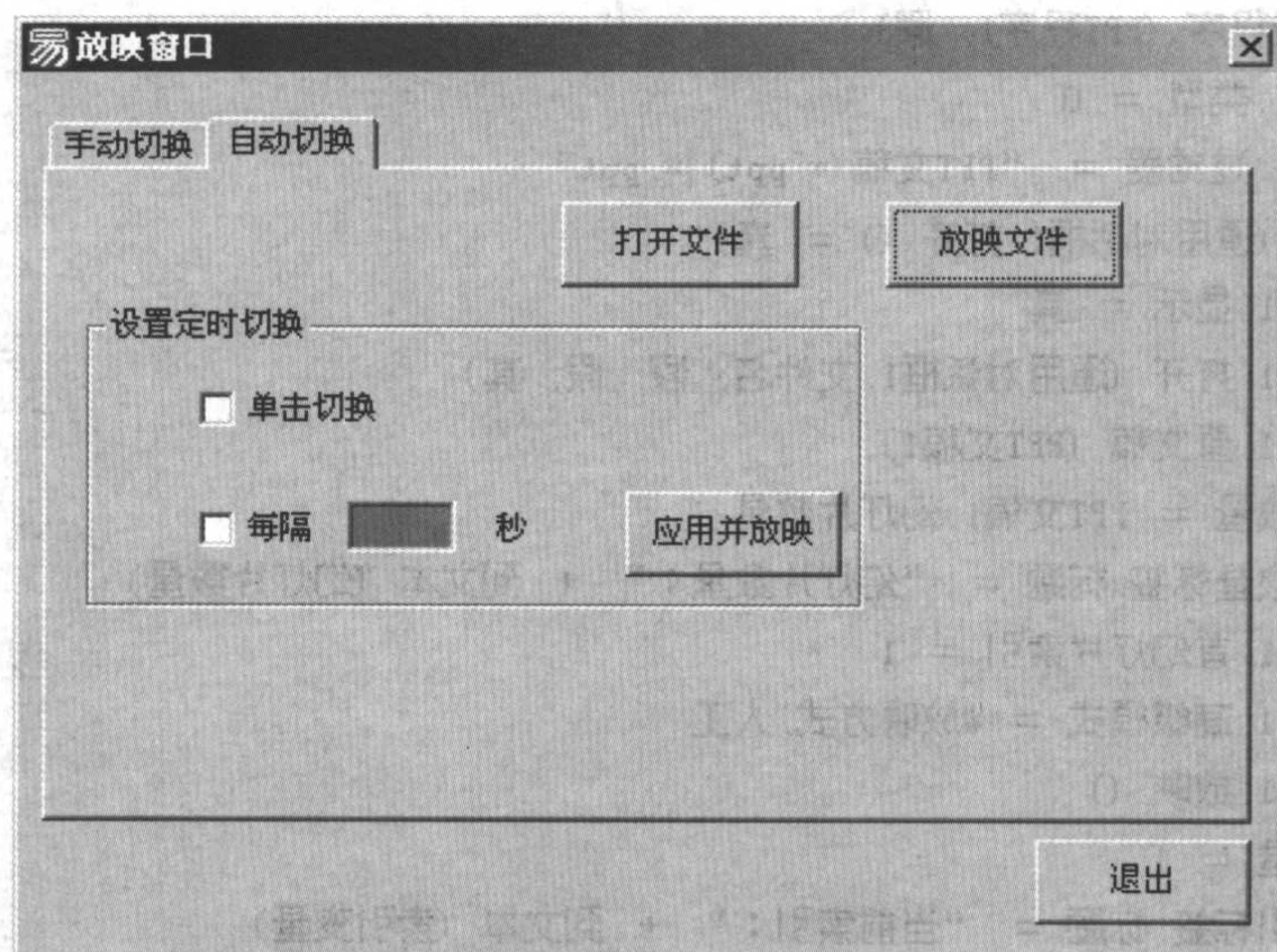


图 29-2 自动切换设置窗口

“\_打开按钮\_被单击”事件子程序的代码如下：

子程序名	返回值类型	公开	备注
_打开按钮_被单击			

```
--- 如果真 (PPT程序1.是否已创建 = 0)
PPT程序1.创建 ()
PPT文稿1.置程序 (PPT程序1, 假)
通用对话框1.类型 = 0
通用对话框1.过滤器 = "PPT文稿 (*.ppt) | *.ppt"
--- 如果真 (通用对话框1.打开 () = 真)
PPT程序1.显示 = 真
PPT文稿1.打开 (通用对话框1.文件名, 假, 假, 真)
PPT播放1.置文稿 (PPT文稿1)
幻灯片数量 = PPT文稿1.幻灯片数量
幻灯片数量标签.标题 = "幻灯片数量：" + 到文本 (幻灯片数量)
```

以上代码与手动切换的打开文件代码差别不大，但并没有设置放映方式和使用“放映 ()”方法。所以文稿只是被显示，不能被马上放映，只有单击放映文件按钮，才会自动切换（如果有幻灯片没有被设置定时切换，幻灯片就会停止播放）。

如果想要成批的设置幻灯片的放映间隔，可以通过循环语句设置所有幻灯片的切换间隔。相关程序代码如下：



子程序名	返回值类型	公开	备注
_应用按钮_被单击			

变量名	类型	静态	数组	备注
局部计次变量	整数型			

---▶ 计次循环首 (幻灯片数量, 局部计次变量)

PPT文稿1. 幻灯片索引 = 局部计次变量

当前索引标签. 标题 = "当前索引:" + 到文本 (局部计次变量)

PPT文稿1. 单击切换 = 单击切换选择框. 选中

PPT文稿1. 定时切换 = 间隔选择框. 选中

--- 如果真 (间隔选择框. 选中 = 真)

PPT文稿1. 切换间隔 = 到数值 (间隔编辑框. 内容)

--- 计次循环尾 0

PPT文稿1. 首幻灯片索引 = 1

--- 如果 (单击切换选择框. 选中 = 真)

PPT播放1. 放映类型 = #动画高级模式. 单击

▶ PPT播放1. 高级模式 = #放映方式. 幻灯片时间

▶ PPT播放1. 放映 0

以上代码使用一个“计次循环”，分别设置每个幻灯片的切换间隔。

放映之前设置首张幻灯片的索引。然后根据是否单击切换，设置不同的放映方法。

**注意：**以上程序使用的是 PowerPoint2003 类型库。PowerPoint2003 类型库名称是 Microsoft PowerPoint 11.0 Object Library 版本 2.8。可以使用易语言提供的类型库和 OCX 组件包装器安装。其位置在系统 (Windows XP) 安装盘下，路径为：

安装盘:\Program Files\Microsoft Office\OFFICE11\msppt.olb。

帮助文件路径为：

安装盘:\Program Files\Microsoft Office\OFFICE11\2052\VBAPP10.CHM。





## 第三十章 办公组件支持库

易语言办公组件支持库提供一套功能强大的通用办公组件，主要功能包括：文字处理、表格编辑、数据处理、辅助管理等。可以使用该组件设计制作各种不同风格的办公软件，以适应不同公司或企业的特殊情况及需要。本章就来介绍办公组件的功能及使用方法。

### 30.1 办公组件简介

办公组件从易语言 3.8 版起提供给用户使用，在易语言的扩展组件箱中可以找到。由于它和易语言有很好的兼容性，这样就不需要再使用类型库对 WORD 进行烦琐的操作了，可以直接使用该组件来设计和编写出不同类型并且功能强大的办公软件。

该组件有 4 个自有属性、65 个方法和 10 个自有事件。另还包括了 5 个对象接口：页面接口、工作表接口、对象接口、表格接口、修订接口。这些接口又提供了众多的方法。

这些接口对象可以通过下列方法获得：“取页面接口()”、“取工作表接口()”、“取对象接口()”、“取表格接口()”、“取修订接口()”。取出的接口对象存放在指定的对象变量中。（关于对象的操作方法，请参看“第十七章 COM 对象”中的相关内容。）

由于本组件提供的方法多达 200 多个，所以本章不能做全面的介绍，本章将选择一些典型的方法来做介绍，然后大家可以根据本章的介绍及易语言的帮助系统，自己学习其他命令的使用方法。

易语言也提供了一个较完整的例程，大家可自行扩充它。

### 30.2 办公组件属性

办公组件的自有属性有 4 个，包括“右键菜单”、“工作表菜单”、“水平标尺”及“垂直标尺”。在了解这些属性前，大家首先来看看办公组件的主界面。

首先新建一个易程序，然后在窗口中添加一个办公组件，运行这个易程序，就可以看到办公组件的主界面了。程序运行后，在主窗口上点击鼠标右键，观察一下。如图 30-1 所示。



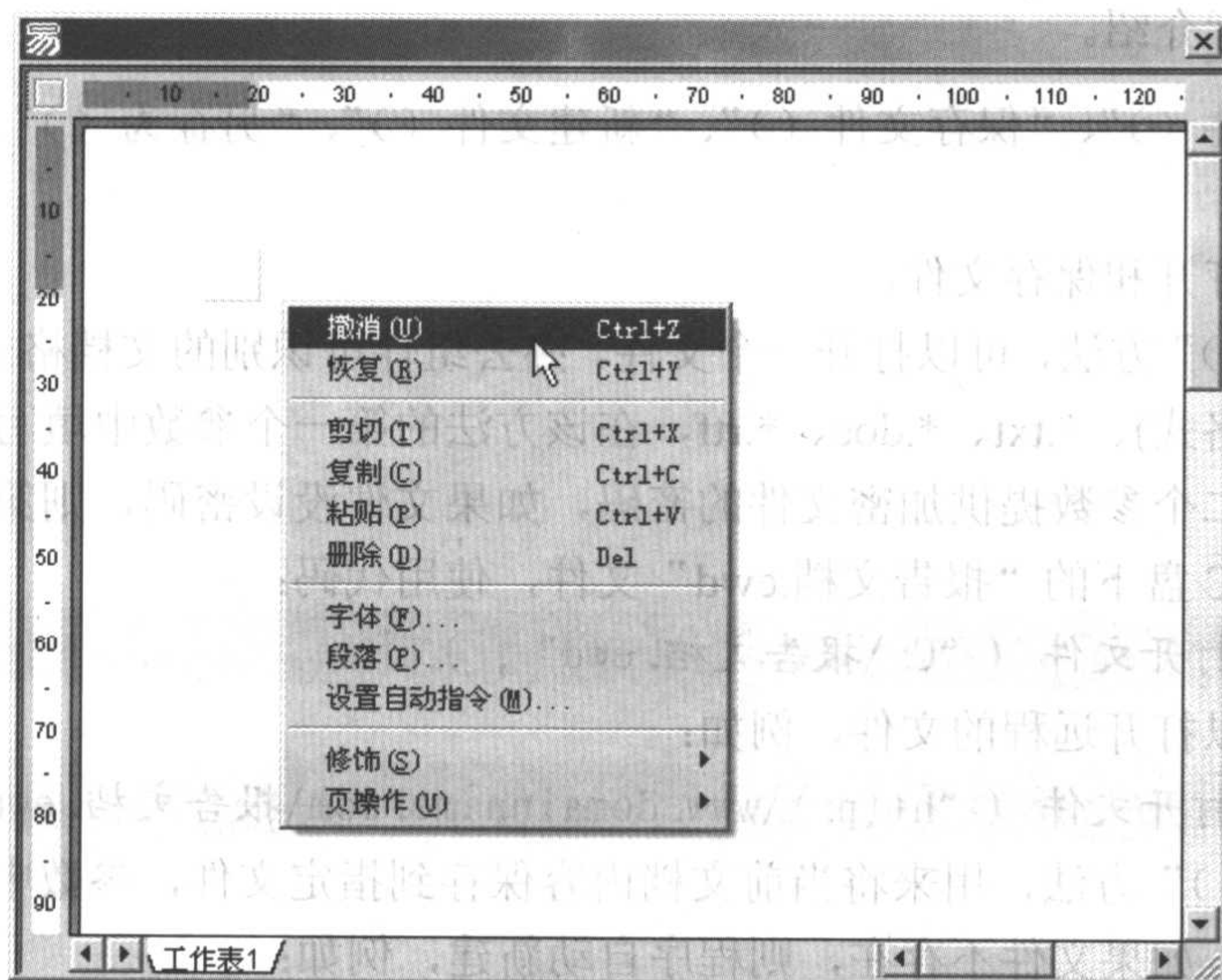


图 30-1 办公组件主界面

办公组件的主界面，中间是编辑区，上边与左侧有横向标尺和纵向标尺。在组件的下方，可以看到有多个工作表的标签，办公组件并且提供强大电子表格功能。

下面来介绍办公组件的属性。

#### 1. “右键菜单”属性

在图 30-1 中可以看到，在办公组件上点击鼠标右键时，会弹出右键菜单，该菜单用来对文档进行撤消、复制、粘贴等基本操作及对文档进行字体、段落等基本设置。

“右键菜单”属性就是控制该菜单是否允许弹出，要想禁止默认的右键菜单，就将本属性设置为“假”。

#### 2. “工作表菜单”属性

当在办公组件的工作表标签上点击鼠标右键时会弹出工作表菜单，该菜单用来进行插入工作表，复制、粘贴和删除工作表等操作。

“工作表菜单”属性用来控制该菜单是否允许被弹出，将“工作表菜单”属性设置为“假”，可以禁止弹出工作表菜单。

#### 3. “水平标尺”和“垂直标尺”属性

逻辑型。这两个属性用来控制办公组件的水平标尺和垂直标尺是否可视。

## 30.3 办公组件的方法

### 30.3.1 办公组件的重要方法

办公组件提供了 65 个专有方法，由于篇幅有限，所以不能进行一一介绍，下面列举一





些典型的方法进行介绍。

### 1. “打开文件 ()”、“保存文件 ()”、“新建文件 ()”、“另存为 ()”、“另存为文本文件 ()”方法

这些方法用打开和保存文件。

“打开文件 ()”方法，可以打开一个文件，办公组件可识别的文档格式包括：\*.ewd(该组件的默认保存格式)、\*.txt、\*.doc、\*.rtf，在该方法的第一个参数中填写要打开文件的全路径文件名，第二个参数提供加密文件的密码，如果文件没设密码，则第二个参数省略不填。例如，打开 C 盘下的“报告文档.ewd”文件，使用代码：

```
办公组件 1. 打开文件 ( “C:\报告文档.ewd” , )
```

本方法还可以打开远程的文件，例如：

```
办公组件 1. 打开文件 ( “http://www.domainname.com\报告文档.ewd” , )
```

“保存文件 ()”方法，用来将当前文档内容保存到指定文件，参数中填写要保存文件的全路径文件名，如果文件不存在，则程序自动新建，例如：

```
办公组件 1. 保存文件 ( “C:\报告文档.ewd” , )
```

“新建文件 ()”方法用来新建一个新文件，同时关闭已打开的文件。该方法没有参数和返回值。

“另存为 ()”方法将当前文件另存为新的文件，参数中提供一个全路径文件名。

“另存为文本 ()”方法将当前文件另存为一个文本文件 (\*.txt 文件)，参数中提供一个全路径文件名。

### 2. “查找 ()”、“替换 ()”、“替换文本 ()”方法

“查找 ()”和“替换 ()”方法运行后，会打开办公组件自带的查找和替换对话框，查找和替换的操作都在相应的对话框中进行。

“替换文本 ()”方法用来替换文档中的指定文本，第一个参数提供欲替换的文本；第二个参数提供替换后的文本；第三个参数为是否区分大小写，是逻辑型参数。

例如将文档中的“中国你好”全部替换成“祖国你好”，使用代码如下：

```
办公组件 1. 替换文本 ( “中国你好” , “祖国你好” , 假)
```

### 3. “复制 ()”、“粘贴 ()”、“剪切 ()”、“撤消 ()”、“重复 ()”、“全选 ()”、“清除 ()”方法

这些方法的功能在办公组件的右键菜单中都有相应的选项，相信熟悉 Windows 基本操作的人应该对这些功能不会陌生，这些方法只是提供在代码中调用这些功能，可以方便在代码中进行批量的操作。

**注意：**“清除 ()”方法相当于右键菜单中的“删除”项的功能，是删除所选区的全部内容，并不是清除文档中的全部内容。

### 4. “取选择区字体 ()”、“置选择区字体 ()”、“字体设置 ()”方法

“取选择区字体 ()”方法用来取出选择区的字体并返回，返回值是一个“字体”类型的，“字体”是易语言核心支持库中提供的数据类型，其中包括成员有“角度”、“加粗”、“倾斜”、“删除线”、“下划线”、“字体大小”、“字体名称”。例如用信息框显示当前选择区



的字体名称，代码如下：

变量名	类型	静态	数组	备注
字体	字体			

字体 = 办公组件1.取选择区字体 ()  
信息框 (字体, 字体名称, 0, )

“置选择区字体 ()”方法用来设置选择区的字体，参数中要提供一个“字体”类型的数据。

“字体设置 ()”方法可以打开办公组件提供的“字体”对话框进行字体设置。该对话框中提供的字体选项很多，所以使用起来也很方便。

对选取操作的方法还包括：“取选择区文本颜色 ()”、“置选择区文本颜色 ()”、“取选择区文本底纹 ()”、“置选择区文本底纹 ()”大家可以参看易语言帮助使用。

### 5. “插入文本 ()”方法

该方法用来在文档的指定位置插入文本。

第一个参数为“对象名”，可选参数，文本型。指定文本插入的到的对象。可以指定文本框或单元格的名称（别名）。如果本参数为空，则默认插入到当前页面。

第二个参数为“插入位置”指定文本插入的位置，可以为“文本位置”枚举常量集中的常量之一。

第三个参数指定欲插入的文本内容。

参数中可以填写“文本位置”枚举常量，该常量中包含 3 个成员：“最前”，常量值为 0；“最后”，常量值为 1；“当前位置”，即当前光标位置，常量值为 3。

例如，将“我爱易语言”这几个字符插入到文档的当前位置，使用代码：

办公组件 1.插入文本 (, #文本位置.当前位置, “我爱易语言”)

### 6. “是否首行缩进 ()”、“置首行缩进 ()”方法

“是否首行缩进 ()”方法可以取出当前段落的首行是否缩进。如果当前段落首行有缩进则返回真，否则返回假。

“置首行缩进 ()”方法设置当前段落首行是否缩进，参数中添入真则首行缩进。

### 7. “段落设置 ()”方法

该方法用来打开办公组件内置的段落设置对话框，可以设置段落的水平和垂直对齐方向、行间距及首行缩进的距离。

### 8. “打印 ()”和“打印预览 ()”方法

“打印 ()”方法用来打印当前文档；

“打印预览 ()”方法用来预览要打印的文档，该方法会调用组件内置的预览窗口。这两个方法直接调用即可，没有返回值和参数。

### 9. “取页面接口 ()”、“取工作表接口 ()”、“取对象接口 ()”、“取表格接口 ()”、“取修订接口 ()”方法





这些方法用来取出办公组件提供的 5 个对象，每种对象都提供了很多方法，来实现对指定对象的操作，可以将取得的对象存放在对象类型的变量中，例如：

变量名	类型	静态	数组	备注
页面对象	页面接口			

页面对象 = 办公组件1.取页面接口 ()

页面对象.取页眉 ()

关于对象提供的方法，下面将会具体介绍。

### 30.3.2 办公组件中接口对象的重要方法

办公组件提供了 5 个接口对象，每个对象都提供了很多方法，下面以“页面接口”和“表格接口”为例做介绍。

#### 一. 页面接口

页面接口提供的方法主要是对文档页面外观进行操作，包括边距、页眉、页脚的操作等等。

##### 1. “是否为空 ()”方法

如果页面接口为空（即没有被赋值），返回真；否则返回假。

注意：如果调用一个空接口的其他成员命令将引发运行时错误。

例如，在运行一个页面接口方法前，先判断当前接口是否为空，代码如下：

变量名	类型	静态	数组	备注
页面对象	页面接口			

页面对象 = 办公组件1.取页面接口 ()

--- 如果真 (页面对象.是否为空 () = 假)

↓  
页面对象.添加页 ()

##### 2. “插入页眉 ()”、“取页眉 ()”、“置页眉 ()”、“删除页眉 ()”方法

这些方法都是对文档的页眉进行操作的方法。

“插入页眉 ()”方法可以用来插入和删除页眉，当前文档如果已有页眉，则该方法运行后会将页眉删除；如果当前文档没有页眉，则该方法运行后会在当前文档中插入页眉。

注意：如果当前文档没有页眉，则该方法运行后，自动进入页眉编辑状态；如果当前文档已有页眉，则自动弹出信息框提示是否删除页眉。

“取页眉 ()”方法可以返回当前文档的页眉文本。

“置页眉 ()”方法用来设置页眉文本，第一个参数指定页眉的文本；第二个参数指定页眉的位置，可以选择“0、左对齐”；“1、居中”；“2、右对齐”。

“删除页眉 ()”方法可以用来删除当前文档的页眉。

##### 3. “取页数 ()”和“置当前页 ()”方法

“取页数 ()”方法可以返回当前文档的总页数。



“置当前页 ()”方法可以设置当前页，参数中添入要设置为当前页的页码。

#### 4. “置背景色 ()”和“置底图 ()”方法

这两个方法可以分别用来设置文档的背景色和底图。

#### 5. “输出页面图像 ()”方法

该方法可以将指定页面输出为一个 bmp 图片，第一个参数指定输入文件的全路径文件名；第二个参数填写要输入的页号；第三个参数为输入的比例，默认是 100，即按照百分之百的比例输入图像；第四个参数为逻辑型，设置是否以镜像图像方式输出。

### 二. 表格接口

该接口提供的方法，用来专门对文档中的表格进行操作。

#### 1. “是否为表格 ()”方法

该方法用来判断当前被选择的对象是否是表格，如果是则返回“真”，否则返回“假”。在对表格进行操作前可以使用该命令首先进行判断，防止程序出现错误。

#### 2. “取行数 ()”和“取列数 ()”方法

这两个方法可以返回指定表格的行数及列数，参数中填写要返回行列数的表格名称。

例如，返回当前被选中表格的行列数，使用代码：

变量名	类型	静态	数组	备注
页面接口	页面接口			
对象接口	对象接口			
表格接口	表格接口			
表格名称	文本型			

页面接口 = 办公组件1.取页面接口 ()

对象接口 = 办公组件1.取对象接口 ()

表格接口 = 办公组件1.取表格接口 ()

对象接口.插入 (#对象种类,表格)

--- 如果真 (表格接口.是否为表格 () = 真)

表格名称 = 对象接口.取当前名称 ()

信息框 (表格接口.取行数 (表格名称), 0, )

信息框 (表格接口.取列数 (表格名称), 0, )

代码中使用了 2 个“对象接口”提供的方法。

“插入 ()”方法，可以在文档中插入指定类型的对象，提供枚举常量“对象种类”中的成员，该枚举常量中包括成员有：圆、圆形印章、圆锥、曲线、直线等等，多达 26 种对象；

“取当前名称 ()”方法，可以取得当前被选中的对象名称。

注意：在“表格接口”的方法中，很多参数要求填写表格名称，都可以使用“表格接口”的“是否为表格 ()”方法，首先判断当前选中的是否是表格对象，然后使用“对象接





口”的“取当前名称()”方法取得被选中表格的名称；并且有“表格接口”中的很多方法是在当前选中对象是表格的时候才有效果，例如：“画表格线()”和“擦除表格线()”方法。

3. “可否删除当前行()”、“可否删除当前列()”、“可否横向分割()”、“可否纵向分割()”等方法

表格接口提供了很多判断当前单元格或表格状态的方法，这些从方法名称上即可以看出这些方法要判断的内容，如“可否删除当前行()”方法用来判断当前行可否被删除，“可否横向分割()”判断可否对当前选中表格进行横向分割。

在程序中，在对表格进行操作前可以先使用这类命令进行判断，确定可以进行该项操作后，在使用相应的方法进行操作。例如，在删除当前行前先使用“可否删除当前行()”方法进行判断，代码如下：

```
如果真 (表格接口.可否删除当前行 () = 真)
    表格接口.删除当前行 ()
```

4. “画表格线()”和“擦除表格线()”方法

“画表格线()”方法，如果当前选中对象是表格，则鼠标指针会变成笔的形状，可以在当前表格中画表格线。

“擦除表格线()”方法，如果当前选中对象是表格，则鼠标指针会变成橡皮擦的形状，可以在当前表格中擦除表格线。

5. “插入行()”和“插入列()”方法

这两个方法可以在选中表格中插入行列，参数中填写“位置”枚举常量集合中的常量值之一，如果省略则向后插入。

例如在当前表格所选行前插入一行，使用代码：

```
如果真 (表格接口.可否向前插入行 () = 真)
    表格接口.插入行 (#位置.之前)
```

6. “拆分单元格()”和“合并单元格()”方法

“拆分单元格()”方法运行后，会弹出拆分单元格对话框，可以设置当前被选中单元格拆分为几行几列。

“合并单元格()”方法可以合并当前被选中的单元格。

7. “添加后选别名()”方法

当在表格中点击鼠标右键，弹出右键菜单，选择菜单中的“设置别名”选项，会打开“设置别名”对话框，用来设置当前单元格的别名，单元格的别名经常用来对指定单元格进行操作，“表格接口”中有一些方法的参数就需要使用到单元格的别名。

“添加后选别名()”方法可以为“设置别名”对话框中的组合框添加备选项目，可以一次添加多个别名，该方法的参数中，将各个别名之间用“|”分割，比如“别名 1|别名 2|



别名 3”。

例如：添加“姓名格”、“年龄格”和“性别格”几个后选别名，使用代码：

表格接口. 添加候选别名 (“姓名格|年龄格|性别格”)

代码运行后，“设置别名”对话框会出现这些备选项，如图 30-2 所示。

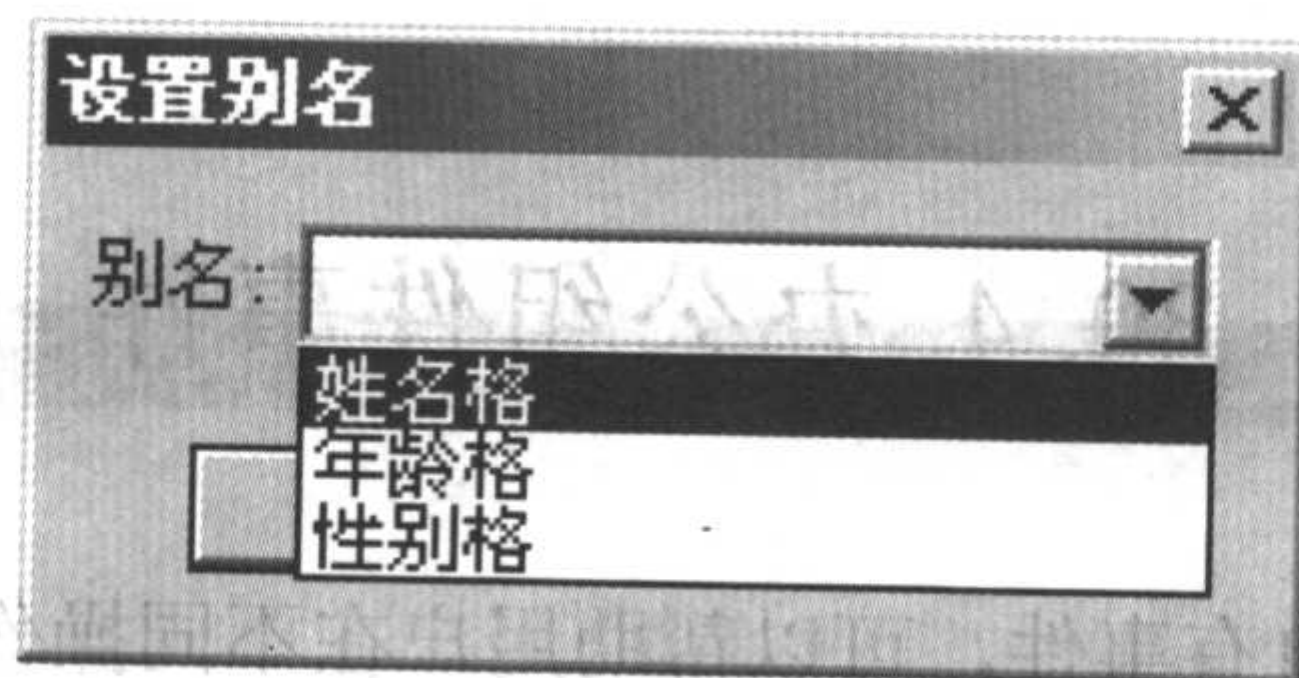


图 30-2 设置别名对话框

#### 8. “取所在行 ()” 和 “取所在列 ()” 方法

“取所在行 ()” 方法返回指定单元格在表格中的行号。

“取所在列 ()” 方法返回指定单元格在表格中的列号。

这两个方法的参数要求填写单元格的别名，如果表格中单元格的别名重复，则返回找到的第一个单元格的行列号。

#### 9. “取单元格文本 ()” 和 “取表格单元格文本 ()” 方法

“取单元格文本 ()” 方法是利用单元格的别名取得指定单元格的文本。

“取表格单元格文本 ()” 方法可以取得某表格指定行列的单元格文本。第一个参数为表格名；第二个参数为指定单元格的行号；第三个参数为指定单元格的列号。

#### 10. “置当前单元格线形 ()” 方法

设置当前单元格线条类型。参数中填写 1~14 数中的任意整数，每个整数代表了 1 种效果，1-14 种单元格线形如图 30-3 所示。

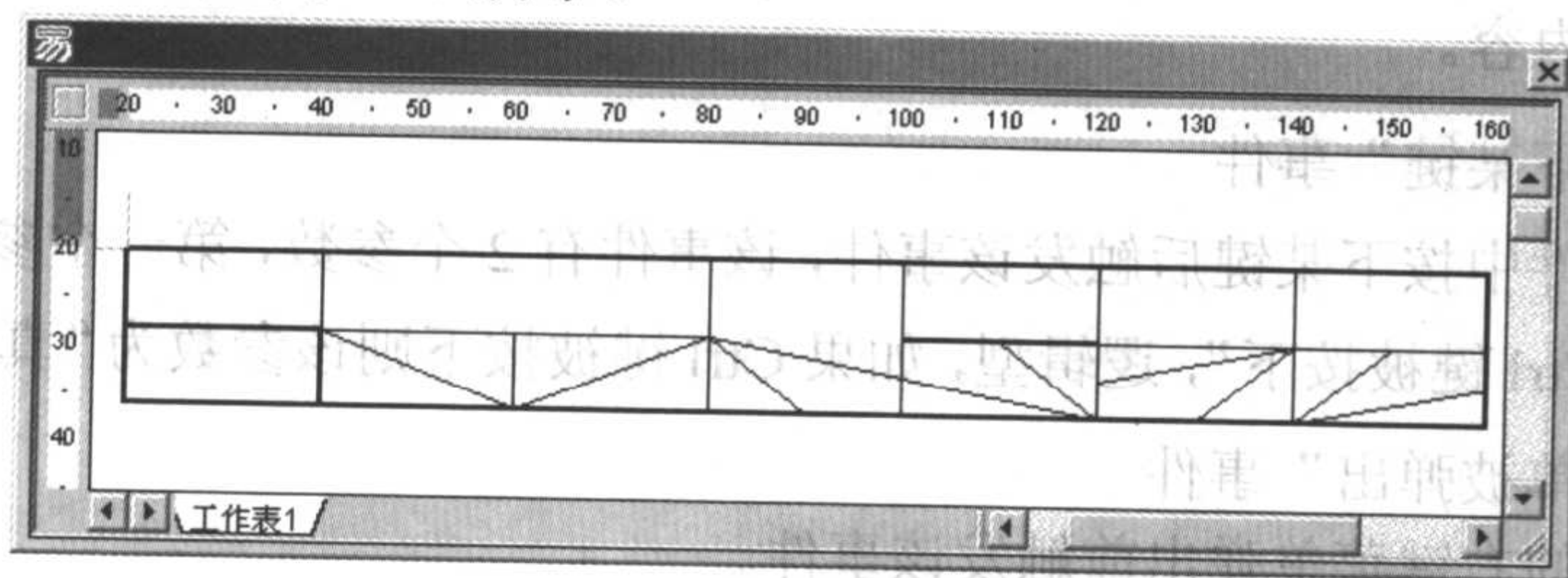


图 30-3 单元格的线条效果

注意：第二次给同一个单元格设置相同类型的线形，则表示删除该线形。

#### 11. “取单元格类型 ()” 和 “置单元格类型 ()” 方法

“取单元格类型 ()” 方法返回指定的单元格的类型。返回值可以为以下常数：0、文本；可输入和文本；2、选择框；3、单选框。参数中添入单元格别名。

“置单元格类型 ()” 方法设置指定的单元格的类型。第一个参数为单元格别名；第二





个参数为欲设置的类型，可以为：0、文本：可输入和文本；2、选择框；3、单选框。

### 12. “输入公式( )”方法

该方法可以打开“单元格公式输入”窗口，可以在该窗口中编辑公式，然后添加到单元格中。

## 30.4 办公组件事件

办公组件提供了 10 种自有事件，可以帮助用户在不同操作发生后进行不同的处理。

### 1. “组件鼠标左键被按下”、“组件被双击”、“组件鼠标左键被放开”事件

和其他组件的鼠标事件类似，办公组件也可以接收鼠标被按下、双击、和放开的事件，但是由于办公组件有右键菜单，所以只接收鼠标左键的事件。

### 2. “即将打开文件”、“打开文件完毕”事件

“即将打开文件”事件在一个新文件打开前触发，可以在该事件中编写提醒用户是否保存等功能。

“打开文件完毕”事件在文件打开完毕后触发。

### 3. “即将保存文件”、“保存文件完毕”事件

“即将保存文件”事件在将组件将要保存时触发。

“保存文件完毕”事件在文件保存完毕后触发。

### 4. “单元格被单击”事件

当表格中的单元格被单击时触发该事件，该事件子程序有 2 个参数，第一个参数为“单元格别名”，是当前被单击的单元格别名；第二个参数为“单元格内容”，是当前被单击的单元格中的文本内容。

### 5. “组件按下某键”事件

当在办公组件中按下某键后触发该事件，该事件有 2 个参数，第一个参数为“键代码”；第二个参数为“Ctrl 键被按下”，逻辑型，如果 Ctrl 键被按下则该参数为“真”，否则为“假”。

### 6. “右键菜单被弹出”事件

当办公组件的右键菜单弹出前触发该事件。

## 30.5 办公组件例程

利用办公组件的强大功能，完全可以开发出功能全面的办公软件，若再加上网络功能，还可以编写强大的网络型办公软件。由于篇幅有限，下面只使用办公组件编写一个小型的



办公软件，来进一步了解办公组件的使用方法。

首先，新建一个易程序，在窗口中添加一个“办公组件”和 2 个通用对话框组件。

然后，进入菜单编辑器，程序中的主要功能都通过菜单的形式来完成。程序界面和菜单项目如图 30-4 所示。

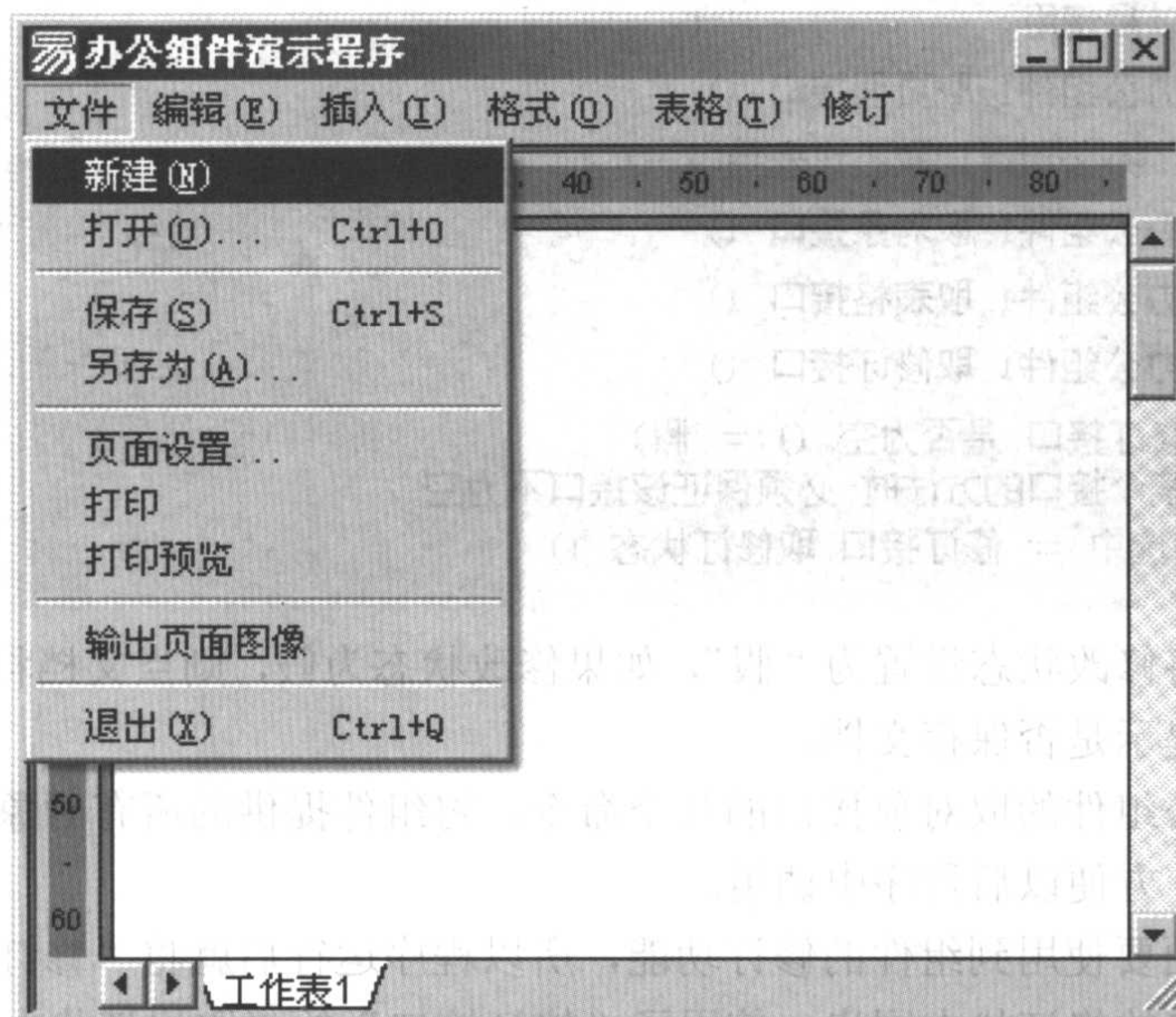


图 30-4 例程界面

下面大家跟着本书一起来完成这些菜单的功能，首先从文件菜单开始，这个菜单下的功能比较简单，办公组件已经提供了很方便的方法，所以很多功能只需要一行代码就可以完成。

首先编写“\_\_启动窗口\_创建完毕”子程序中的代码，在该子程序中做一些初始化程序的工作，如添加一些程序集变量：

窗口程序集名	保留	备注	
窗口程序集1			
变量名	类型	数组	备注
页面接口	页面接口		
工作表接口	工作表接口		
对象接口	对象接口		
表格接口	表格接口		
修订接口	修订接口		

以上是程序中使用到的程序集变量。下面是在启动窗口创建完毕事件中添加代码：





子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

办公组件1. 置修改状态 (假)

默认文档没有被修改

取出接口供以后使用

页面接口 = 办公组件1. 取页面接口 ()

工作表接口 = 办公组件1. 取工作表接口 ()

对象接口 = 办公组件1. 取对象接口 ()

表格接口 = 办公组件1. 取表格接口 ()

修订接口 = 办公组件1. 取修订接口 ()

如果真 (修订接口. 是否为空 () = 假)

在调用某个接口的方法时, 必须保证该接口不为空

修订状态. 选中 = 修订接口. 取修订状态 ()

程序中首先将修改状态设置为“假”，如果修改状态为假，则当文档将被关闭时，会自动弹出信息框，提示是否保存文档。

然后使用办公组件的取对象接口的几个命令，将组件提供的所有对象接口取出并存放在程序集变量中，方便以后程序中调用。

由于程序后面要使用到组件的修订功能，所以程序运行后就将“修订状态”菜单的选中状态和修订接口的修订状态同步，使用了“修订接口”的“取修订状态 ()”方法。

“文件”菜单中“新建”和“打开”子菜单项的代码如下：

子程序名	返回值类型	公开	备注
_新建_被选择			

办公组件1. 新建文件 ()

子程序名	返回值类型	公开	备注
_打开_被选择			

通用对话框1. 类型 = 0

通用对话框1. 过滤器 = “易语言办公组件文件 (\*. ewd) | \*. ewd | OST文件 (\*. ost) | 文本文件 (\*. txt) | \*. txt”

通用对话框1. 标题 = “打开”

如果真 (通用对话框1. 打开 ())

办公组件1. 打开文件 (通用对话框1. 文件名, )

办公组件1. 置修改状态 (假)

新建文档的功能直接使用了办公组件的“新建文件 ()”方法，如果原先打开编辑的文档被修改过，则自动弹出信息框提示是否保存旧文档。

“打开”菜单被选择后，则首先将“通用对话框1”的类型设置为0，即打开文件通用对话框，然后设置通用对话框的过滤器为办公组件可以识别的几种文件类型。

然后编写“保存”和“另存为”菜单的功能，“保存”功能，如果当前文档未被保存过，则弹出通用对话框保存到新文件，如果当前文档被保存过或当前文档是打开的旧文档，则



保存文档到旧的文档中。“另存为”功能，则不论当前文档是否被保存过，都保存到新的文件中。代码如下：

子程序名	返回值类型	公开	备注
_保存_被选择			

如果 (办公组件1.取文件路径 () ≠ “”)  
    不是第一次保存  
    办公组件1.保存文件 (办公组件1.取文件路径 ())  
    办公组件1.置修改状态 (假)  
    \_另存为\_被选择 () ‘第一次保存，调用另存为’

---

子程序名	返回值类型	公开	备注
_另存为_被选择			

通用对话框1.类型 = 1  
通用对话框1.过滤器 = “易语言办公组件文件 (\*.ewd)|\*.ewd|OST文件 (\*.ost)”  
通用对话框1.标题 = “另存为”  
--- 如果真 (通用对话框1.打开 ())  
    办公组件1.另存为 (通用对话框1.文件名)  
    办公组件1.置修改状态 (假)

程序中首先使用了办公组件的“取文件路径 ()”方法，如果取出的文件路径不为空，表示不是第一次保存，则将当前文档保存到旧的文档中。然后将修改状态设置为假。如果取出的文件路径为空，则运行“\_另存为\_被选择”子程序。

“另存为”菜单被选择后，则将通用对话框的类型设置为 1，即保存文件类型，弹出通用对话框，将文件保存到通用对话框设置的文件中。

下面，大家来看“页面设置”、“打印”、“打印预览”、“输入页面图像”和“退出”菜单的代码：

子程序名	返回值类型	公开	备注
_页面设置_被选择			

页面接口.页面设置 () ‘调用页面接口的方法’

子程序名	返回值类型	公开	备注
_打印_被选择			

办公组件1.打印 ()

---

子程序名	返回值类型	公开	备注
_打印预览_被选择			

办公组件1.打印预览 ()





子程序名	返回值类型	公开	备注
_输出页面图像_被选择			

通用对话框1.类型 = 1

通用对话框1.过滤器 = "BMP文件|\*.bmp"

--- 如果真 (通用对话框1.打开 ())

页面接口.输出页面图像 (通用对话框1.文件名, 0, 100, 假)

子程序名	返回值类型	公开	备注
_退出_被选择			

结束 ()

程序中，这些菜单的功能，基本都使用了办公组件和其提供接口的一个方法，就实现了菜单的功能。

下面是“编辑”菜单中各选项的功能，菜单子项如图 30-5 所示。

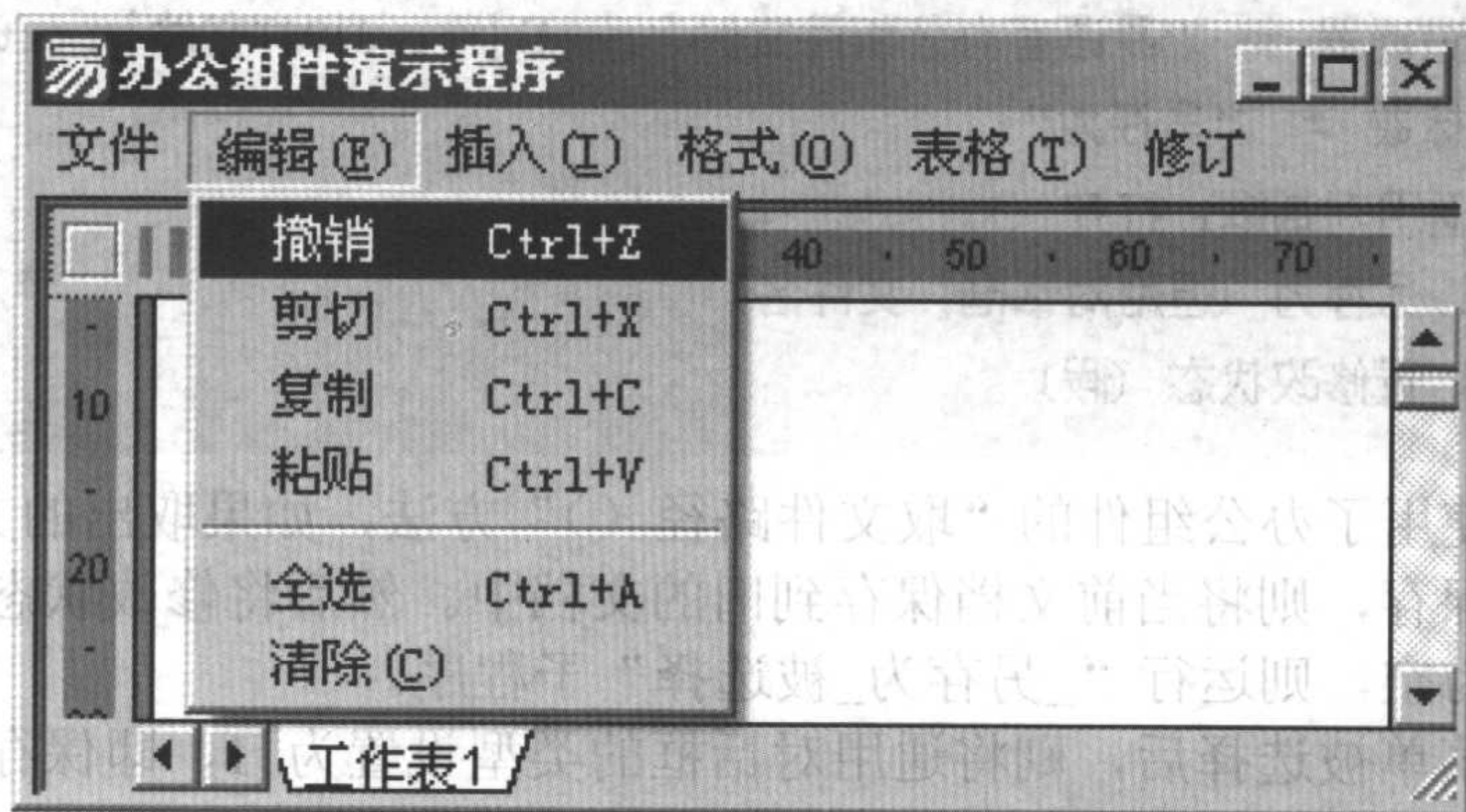


图 30-5 编辑菜单选项

“编辑”菜单中选项的功能，办公组件都提供了方法，程序中可以很方便地实现，代码如下：

子程序名	返回值类型	公开	备注
_撤销_被选择			

办公组件1.撤销 ()

子程序名	返回值类型	公开	备注
_剪切_被选择			

办公组件1.剪切 ()

子程序名	返回值类型	公开	备注
_复制_被选择			



办公组件1. 复制 0

子程序名	返回值类型	公开	备注
_粘贴_被选择			

办公组件1. 粘贴 0

子程序名	返回值类型	公开	备注
_全选_被选择			

办公组件1. 全选 0

子程序名	返回值类型	公开	备注
_清除_被选择			

办公组件1. 清除 0

上述菜单的功能直接调用了办公组件提供的方法，是不是非常方便呀。

接下来，可以编写“插入”菜单下选项的功能了，“插入”菜单选项如图 30-6 所示。

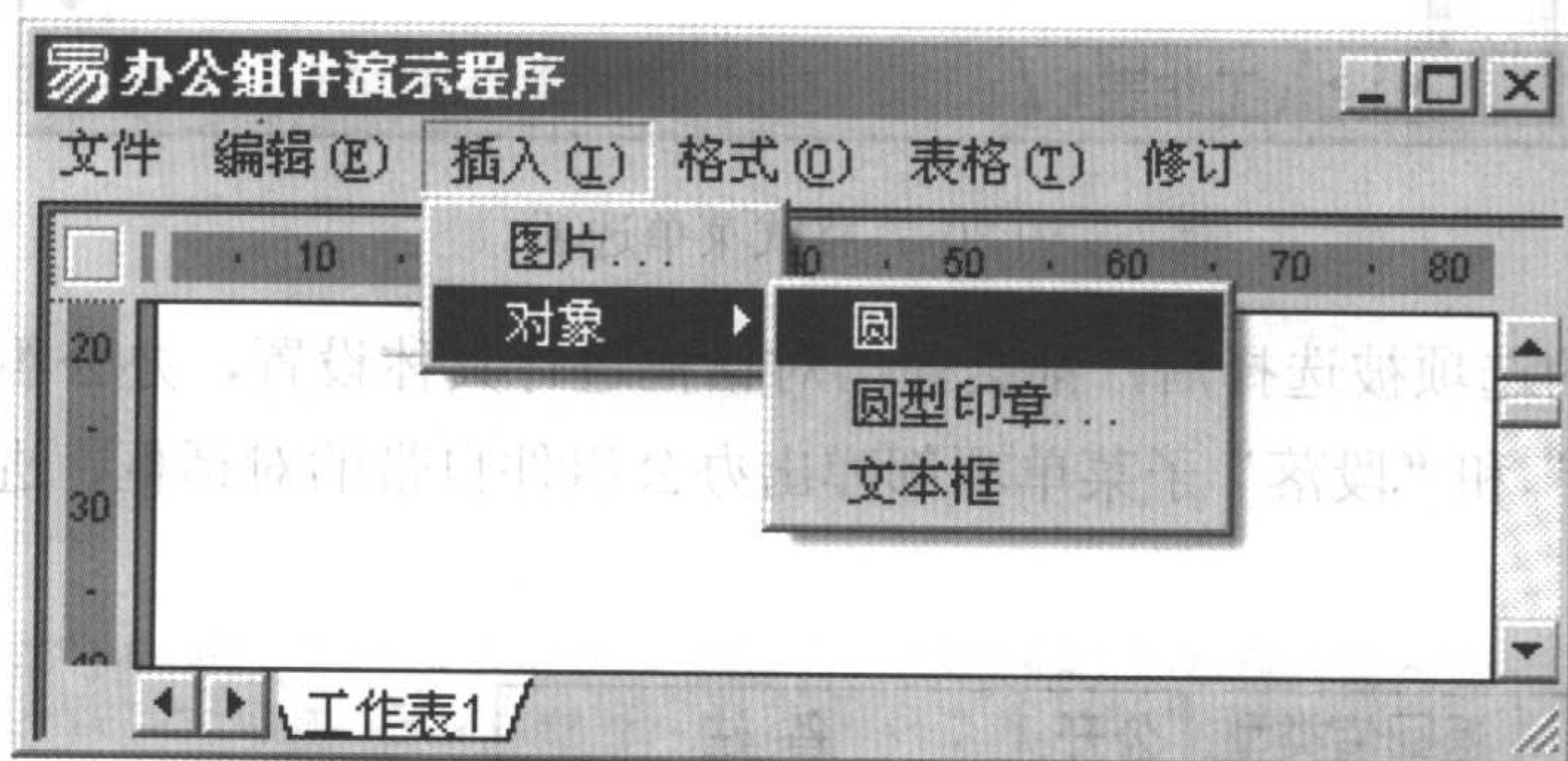


图 30-6 插入菜单选项

这些菜单的功能，办公组件也提供很方便的方法，代码如下：

子程序名	返回值类型	公开	备注
_图片_被选择			

对象接口. 插入 (#对象种类. 图片)

子程序名	返回值类型	公开	备注
_圆_被选择			

对象接口. 插入 (#对象种类. 圆)

子程序名	返回值类型	公开	备注
_圆型印章_被选择			

对象接口. 插入 (#对象种类. 圆型印章)





子程序名	返回值类型	公开	备注
_文本框_被选择			

对象接口.插入 (#对象种类.文本框)

“插入”菜单的功能，都使用了“对象接口”的“插入（）”方法，在参数提供了“对象种类”枚举常量中的常量值。“插入（）”方法还可以插入其他各种对象，大家可以查看易语言帮助中关于“对象种类”枚举常量的介绍。

程序中，“格式”菜单用来设置段落格式和背景图片，菜单选项如图 30-7 所示。

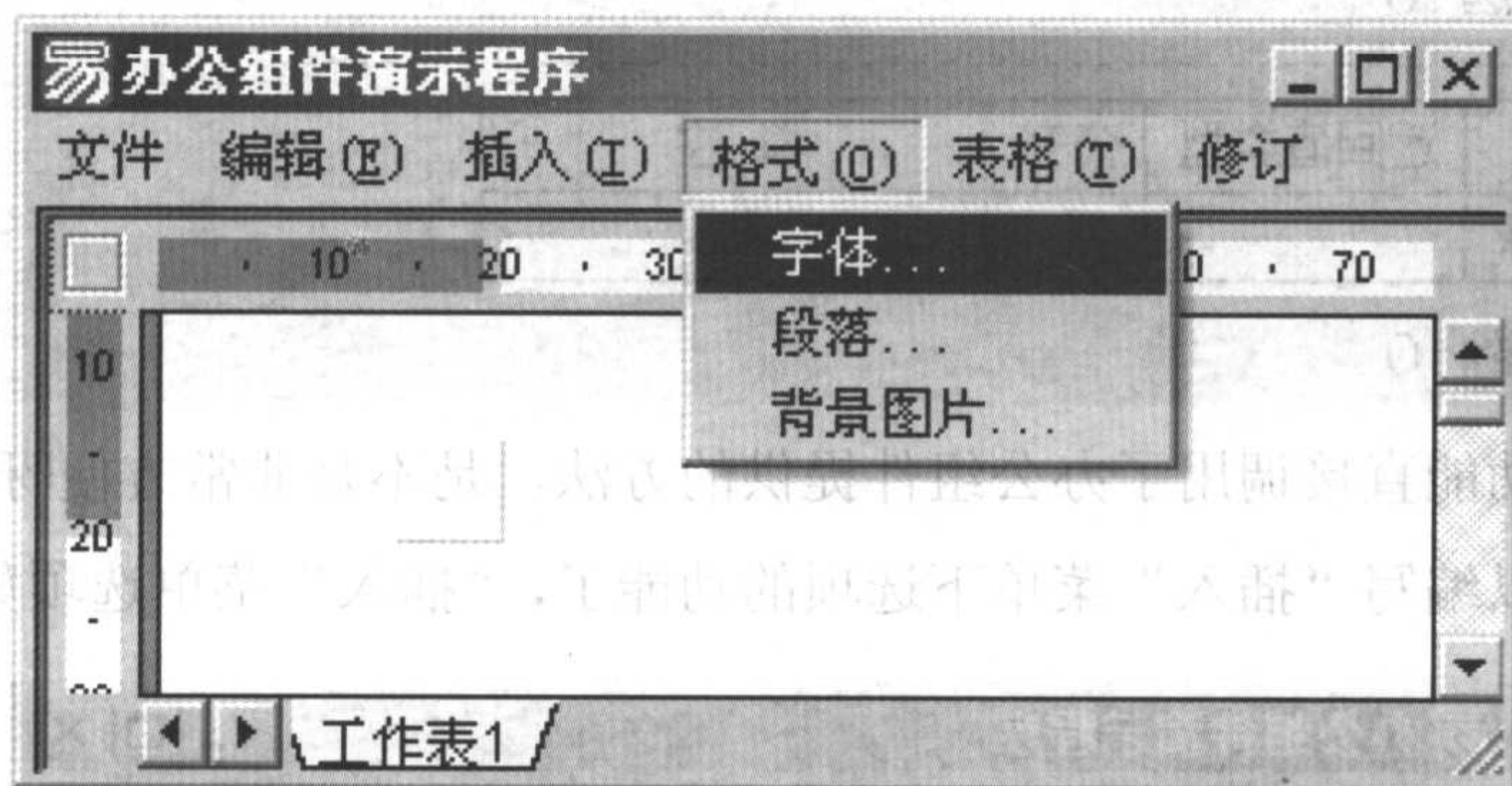


图 30-7 格式菜单选项

格式菜单中的选项被选择后，都会弹出对话框进行具体设置，办公组件内置了很多设置对话框。“字体”和“段落”子菜单项都弹出办公组件自带的对话框，进行字体和段落设置。

子程序名	返回值类型	公开	备注
_字体_被选择			

办公组件1.字体设置 0

子程序名	返回值类型	公开	备注
_段落_被选择			

办公组件1.段落设置 0

子程序名	返回值类型	公开	备注
_背景图片_被选择			

通用对话框1.类型 = 0

通用对话框1.过滤器 = “位图文件 (\*.BMP) | \*.BMP”

通用对话框1.标题 = “打开”

--- 如果真 (通用对话框1.打开 ())

↓ 页面接口.置底图 (通用对话框1.文件名, 1)



程序中，使用办公组件“字体设置（）”和“段落设置（）”方法，运行后会自动弹出对话框，自动将新的设置应用到文档中的选择区中。

“背景图片”选项被选择，则先将通用对话框类型设置为打开文件类型，然后将过滤器设置为“\*.bmp”，然后弹出通用对话框，使用“置底图（）”方法将通用对话框打开的图片设置为文档底图。

下面是“表格”菜单中各选项的功能，“表格”菜单选项如图 30-8 所示。

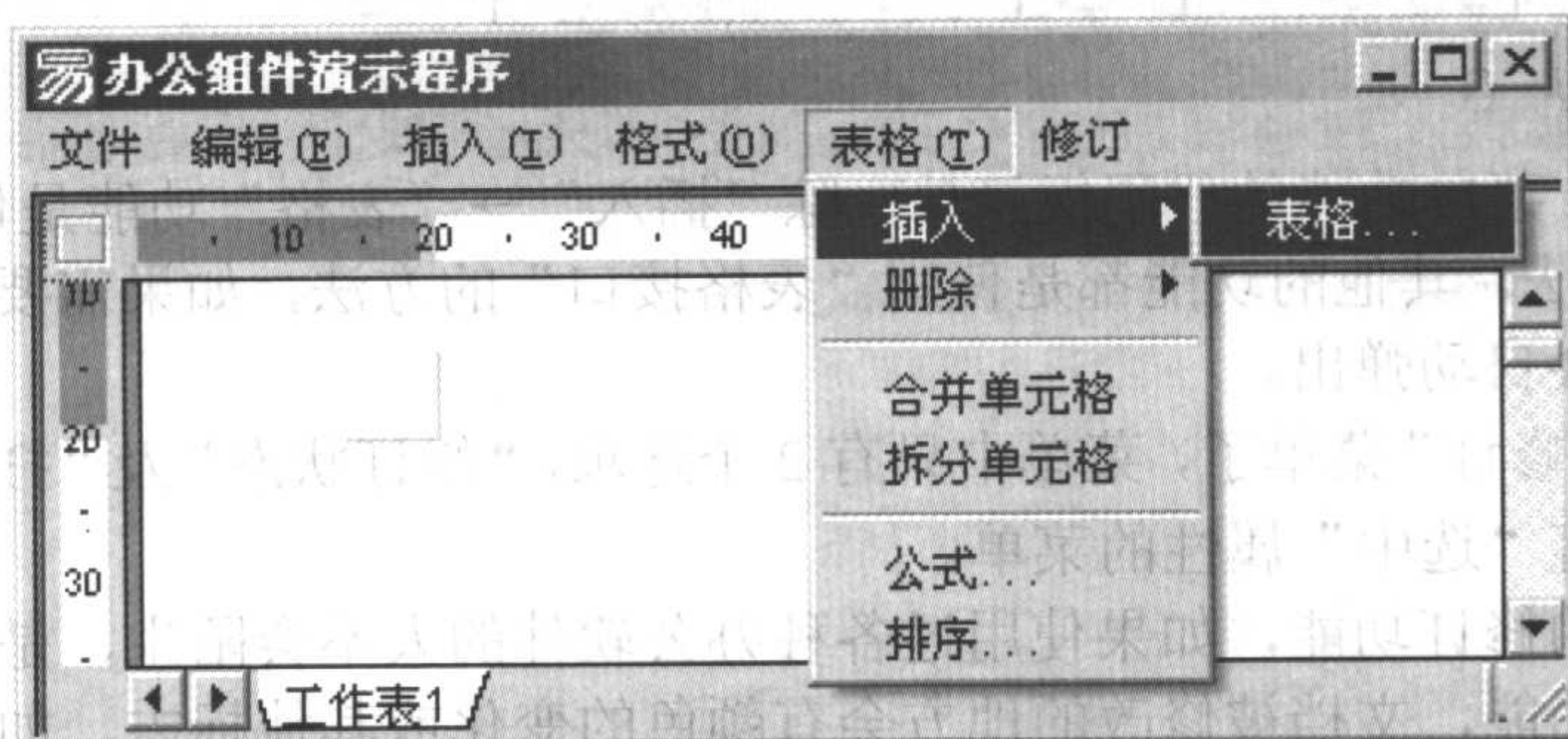


图 30-8 表格菜单选项

图 30-8 中未显示的“删除”菜单的子项中有“当前行”和“当前列”选项，用来删除当前行或列。

实现“表格”菜单中各选项功能的代码如下：

子程序名	返回值类型	公开	备注
_表格_被选择			

对象接口.插入（#对象种类.表格）

子程序名	返回值类型	公开	备注
_当前行_被选择			

表格接口.删除当前行（）

子程序名	返回值类型	公开	备注
_当前列_被选择			

表格接口.删除当前列（）

子程序名	返回值类型	公开	备注
_合并单元格_被选择			

表格接口.合并单元格（）

子程序名	返回值类型	公开	备注
_拆分单元格_被选择			

表格接口.拆分单元格（）





子程序名	返回值类型	公开	备注
_公式_被选择			

表格接口.输入公式 ()

子程序名	返回值类型	公开	备注
_排序_被选择			

表格接口.排序 ()

实现“表格”菜单功能的代码很简单，除“插入”→“表格”功能是使用“对象接口”的“插入 ()”方法，其他的功能都是使用“表格接口”的方法，如果需要弹出对话框和窗口的选项，组件会自动弹出。

最后，只剩“修订”菜单了，菜单中只有 2 个选项，“修订状态”及“打开修订对话框”，“修订状态”是有“选中”属性的菜单。

**提示：**文档的修订功能，如果使用过各种办公软件的人不会陌生，是提供记录当前文档被修改历史的功能，文档被修改的地方会有颜色的变化或其他标志，如果同意对某处的修改，则该处将永久的被修改；如果拒绝文档中某处的修改，则会恢复到修改前的样子。如图 30-9 所示。

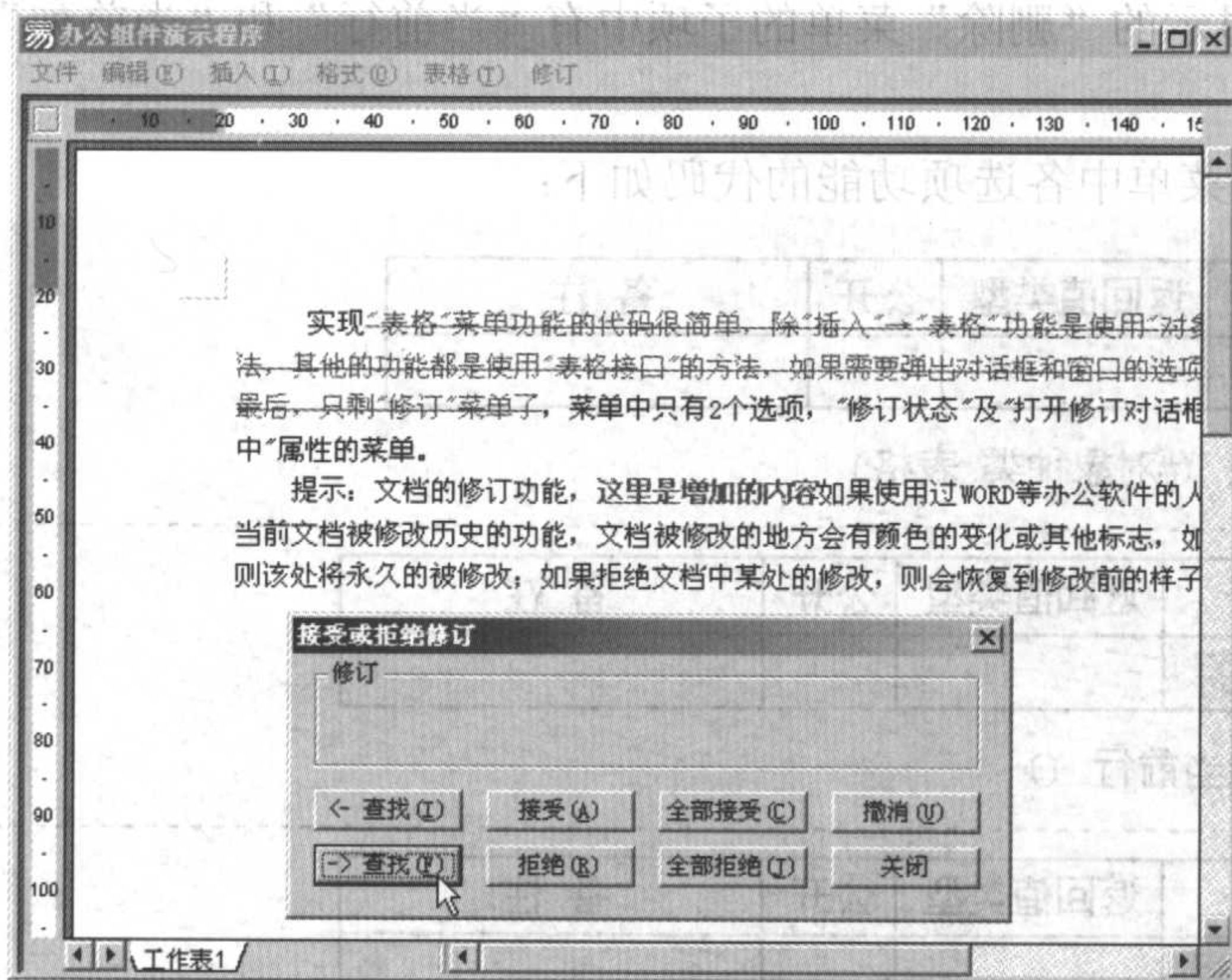


图 30-8 修订状态

修订功能也容易实现，办公组件自带了修订窗口，可以进行文档的修订，代码如下：

子程序名	返回值类型	公开	备注
_修订状态_被选择			

```
如果 (修订状态.选中)
    修订状态.选中 = 假
    修订接口.置修订状态 (假)
    修订状态.选中 = 真
    修订接口.置修订状态 (真)
```



子程序名	返回值类型	公开	备注
_打开修订对话框_被选择			

修订接口.显示对话框 ()

当“修订状态”菜单被选择，则改变其选中状态，同时改变“修订接口”的修订状态，注意：文档的修订功能，只有在修订状态为“真”的时候才会被开启。

“修订接口”的“显示对话框 ()”方法，可以打开修订对话框。

通过以上的操作，一个简单的办公软件就基本成型了，大家会发现，所需要实现的代码非常简单，只要调用办公组件提供的简单的方法，就可以实现强大的功能，易语言用户开发适合自己办公环境的办公软件再也不是麻烦的事情了。

本章节的例程参见随书光盘中的“办公组件例程.e”。

## 30.6 本章小结

办公软件的编写从来就不是一项轻松的工作，如何实现强大的功能和方便的界面设计，工作强度大，也十分烦琐。通过易语言所提供的办公组件，可以非常方便地实现办公类软件的编写，是中国用户的一大福音。办公组件所提供的功能接口简单实用，符合中国人的使用习惯。大家可以加入自己特色的功能，如：带加密功能的办公系统、网络型的办公系统，网络型的绩效考核系统等等，都可轻松实现。

大家还可以进行以下练习：

1. 完善上述办公软件。增加菜单并写出程序代码。
2. 完善上述办公软件。增加工具栏，将常用命令都列在工具栏上，编写相关实现的程序代码。
3. 完善上述办公软件。试增加网络组件，实现局域网内办公传送文件的功能。
4. 办公组件具有打开因特网上文件的功能，试将上述办公软件改写成网络管理形式。





## 附录一 程序调试

几乎每一个稍微复杂一点的程序都必须经过反复的调试、修改，才能最终完成。很显然，只有正确找出错误的地方才可以将其改正。出错以后怎样找出错误的地方就变得很重要了，下面就谈一些查错的方法。

### 调试工具

#### 1. 易语言内部侦错

输入程序代码时有明显的语法错误存在，易语言会提示错误原因，而且当使用 Ctrl+Enter 键编译当前行时，错误代码无法通过编译。当用户写完一段代码后，试运行程序，易语言会对所有可能执行到的语句进行初步编译，当编译时发现明显的格式或语法错误时，易语言会将光标自动移动到错误行，并提示错误原因，用户可以根据光标提示和错误原因很快改正错误。如图 1 所示。

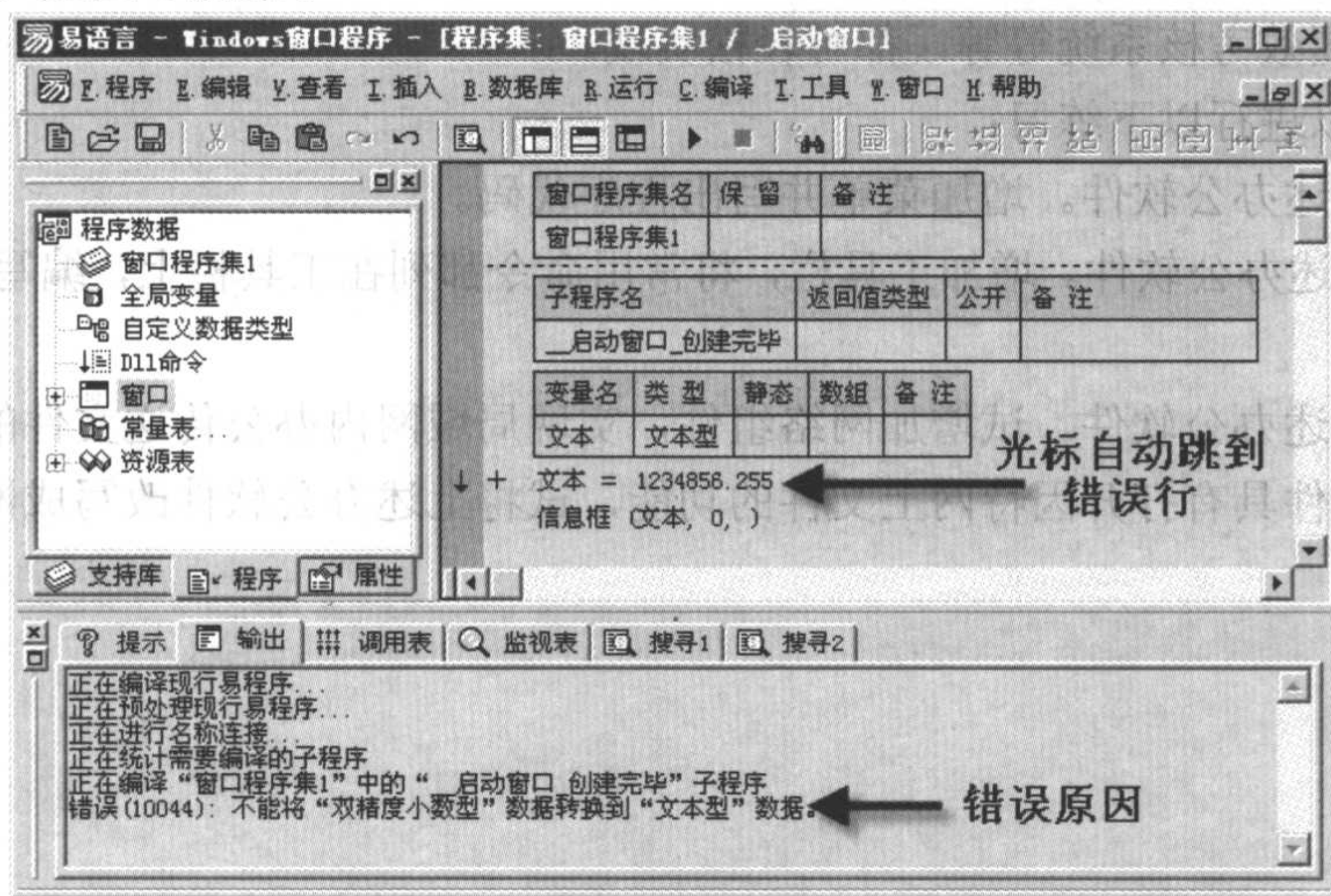


图 1 内部侦错

虽然易语言提供的内部侦错功能十分强大，但也只能指出输入上的格式错误，如数据类型不匹配、指定的变量或组件未找到等。而对于那些程序员编程时，逻辑上的错误，易语言就无能为力了。这就需要其他调试方法和调试命令的配合使用，快速而准确的找到错误代码。



## 2. 跟踪调试方法解释

易语言提供的跟踪调试方法，如图 2 所示。

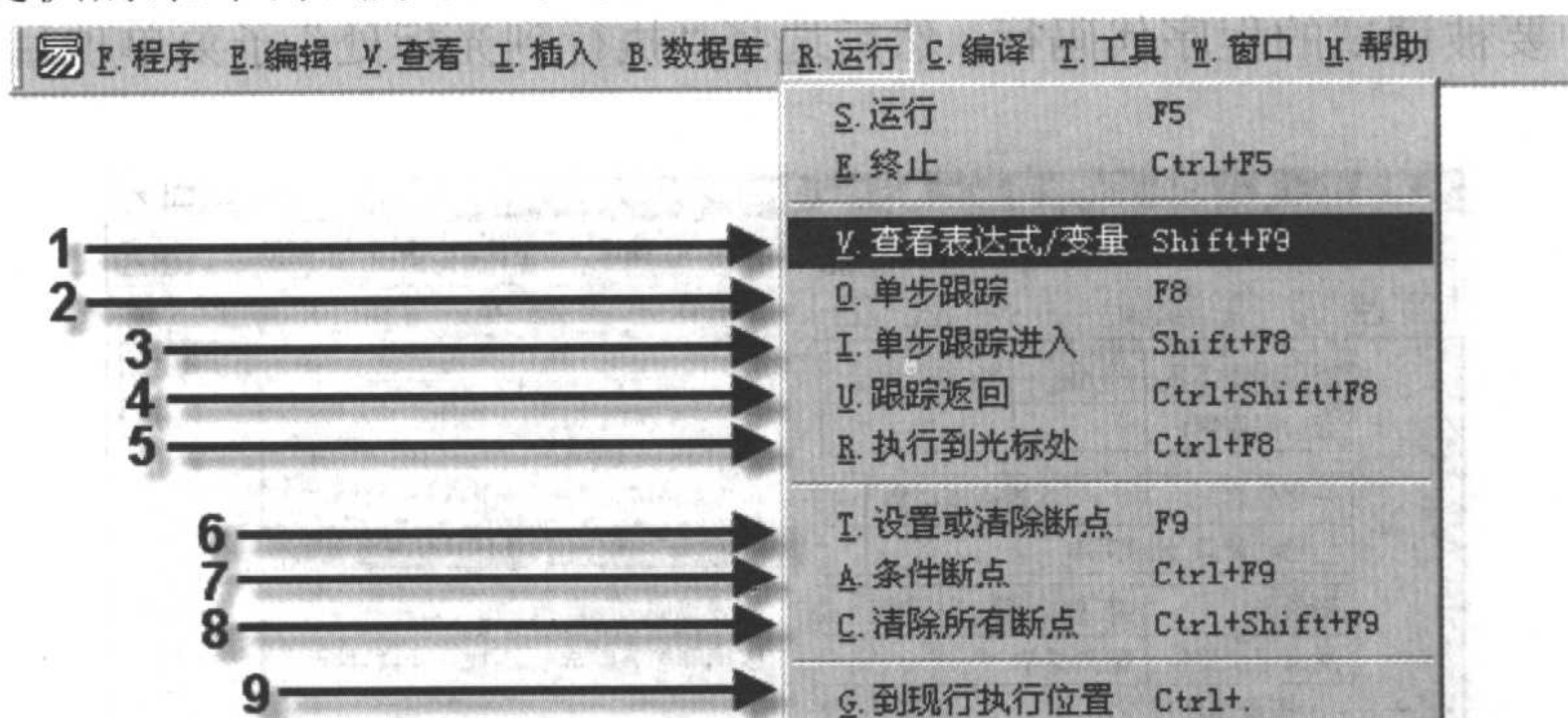


图 2 易语言调试方法

各调试项目在易语言状态条中的解释如下：

- ①在调试过程中，查看指定变量的值，并可以加入到监视表中（监视表位于状态栏中）。
- ②在程序现行运行位置单步执行一程序行，如果此程序行调用了子程序，系统不会跟踪到该子程序中去。

- ③在程序现行运行位置单步执行一程序行，如果此程序行调用了子程序，则跟踪进入子程序。

- ④在上级子程序调用现行子程序的语句后中断。

在被调用子程序中设置断点，断点行被跟踪后，跳出当前子程序，系统跟踪至上级程序调用当前子程序语句的下一行代码。注意：被调用子程序断点后的代码被执行。

- ⑤运行易程序，在当前光标处程序行处中断。

- ⑥设置或清除当前程序行处的断点。

- ⑦设置条件断点，即在断点中加入逻辑判断，当符合条件时，断点才生效。

- ⑧清除掉程序中的所有断点。

- ⑨跳到现行即将被执行语句的位置。

## 3. 跟踪法

编写一个比较简单的程序，看看程序是如何调试的。例程“调试程序.e”代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
局部计次变量	整数型			

```

--> 变量循环首 (1, 300, 1, 局部计次变量)
    -- 如果真 (局部计次变量 % 2 = 0)
        -- 如果真 (局部计次变量 % 3 = 0)
            -- 如果真 (局部计次变量 % 5 = 0)
                编辑框1.内容 = 编辑框1.内容 + 到文本 (局部计次变量) + #换行符
            -- 变量循环尾 0
        -- 变量循环尾 0
    -- 变量循环尾 0
    
```





该程序是输出 300 以内同时能被 2, 3, 5 整除的整数。现在开始调试。调试有多种方法, 先介绍一种不需要设置断点就可以调试程序的方法。

首先选中要被调试的程序代码行。然后选择“执行到光标处”子菜单项目。如图 3 所示。

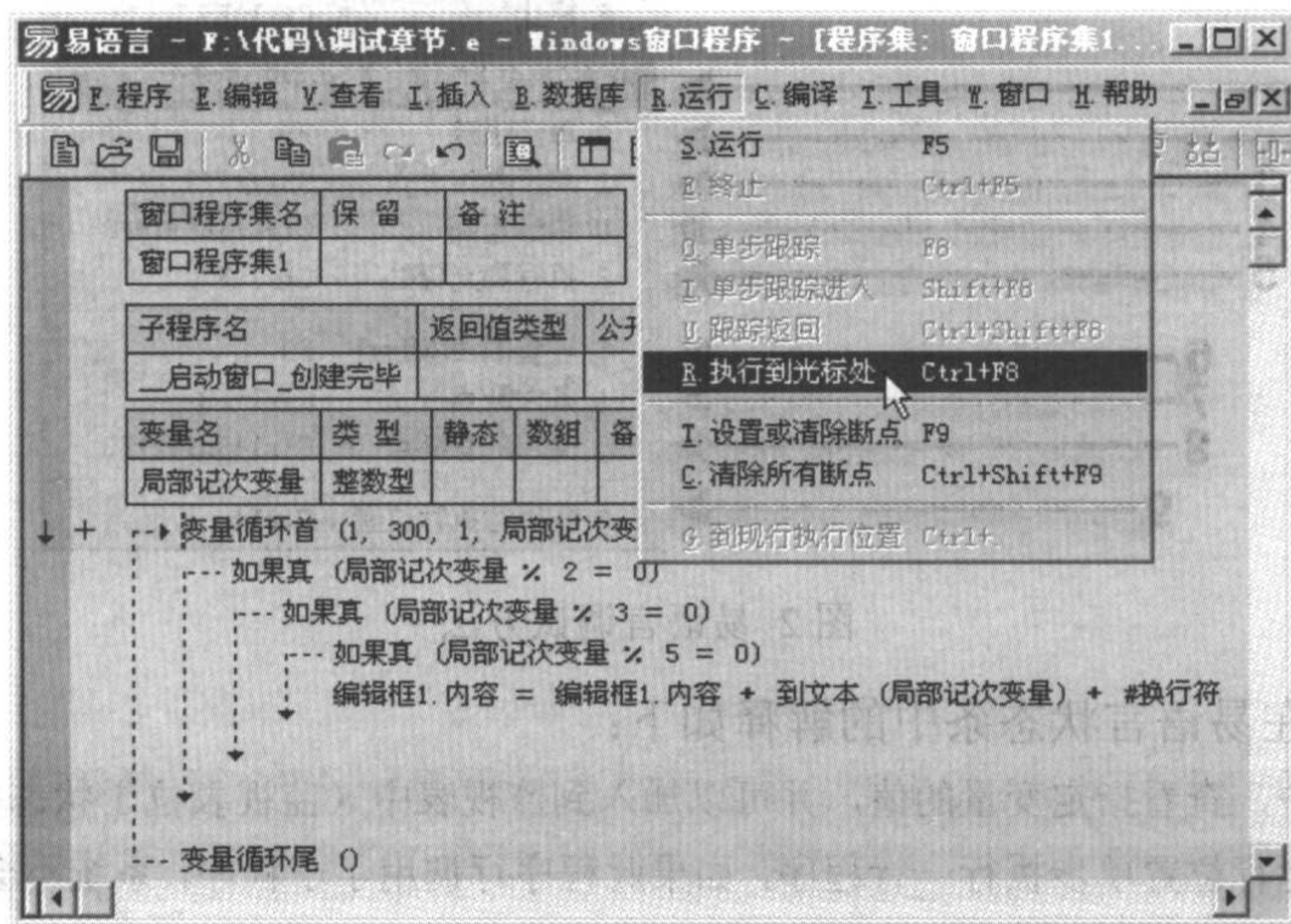


图 3 程序跟踪

按“F5”键运行程序, 当程序执行到被选中代码时, 系统便将程序挂起, 并用黄色箭头指示代码执行位置。

↓ + --> 变量循环首 (1, 300, 1, 局部计次变量)

通常用“F8”键就可以实现把程序每一步执行的情况都反映出来的功能。此方法就是模拟计算机一步步执行程序的过程。

当前程序已经被运行, 不断按“F8”就可以使程序一步一步执行, 直到最后一行代码被执行完毕。

### 3. 调试过程中的变量值查看

编写程序时, 比较容易犯的逻辑错误就是变量赋值错误, 而且变量值的错误不容易查找, 为了方便用户在程序调试的过程中, 更有效的控制和监视变量值的变化过程, 所以易语言加入了“监视表”功能, 可以同时监视多个变量值在程序运行过程中的变化。监视表如图 4 所示。

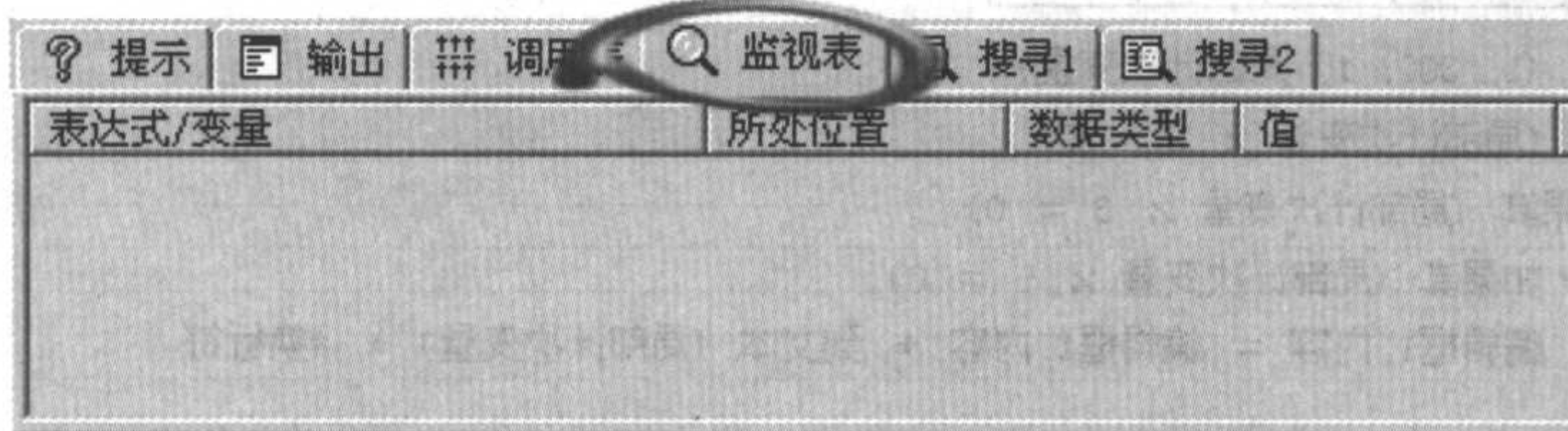


图 4 状态栏中的监视表

首先说明一下使用监视表的几个注意事项。



- 第一、监视表中监视的变量必须是在调试程序被断点暂停的时候才能够显示其值。
- 第二、监视表中监视的变量必须在程序被断点中断时，通过 Shift+F9 键调出查改变量对话框，在对话框中输入需要监控的变量名，然后点击“加入系统监视表”按钮，将指定变量添加到监视表中，如图 5 所示。

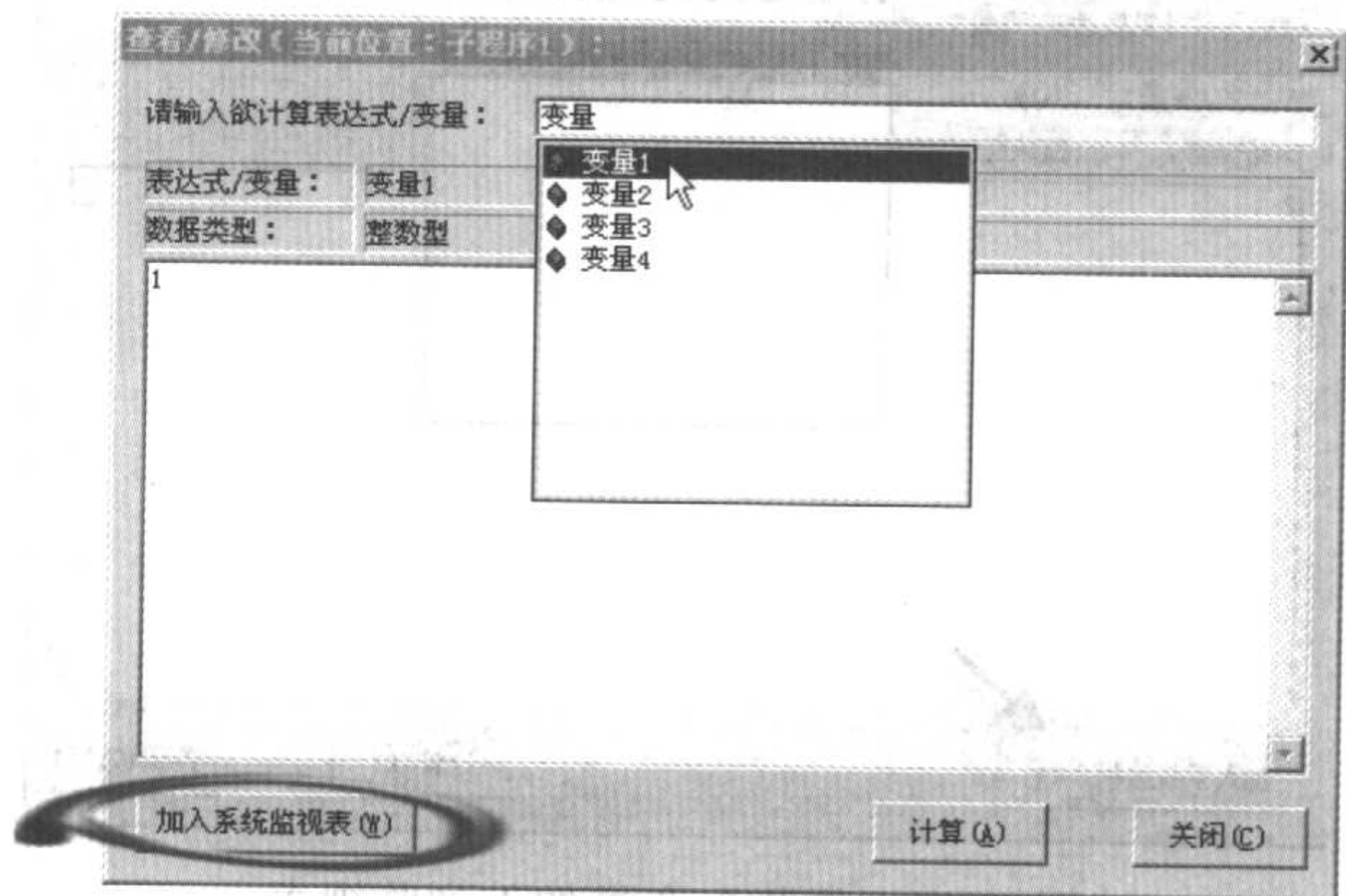


图 5 查改变量对话框

- 第三、查改变量对话框可以查找和更改子程序、程序集中的变量以及全局变量。更改当前监视的子程序可以通过在“调用表”面板中所列出的子程序表中双击选择。如图 6 所示。

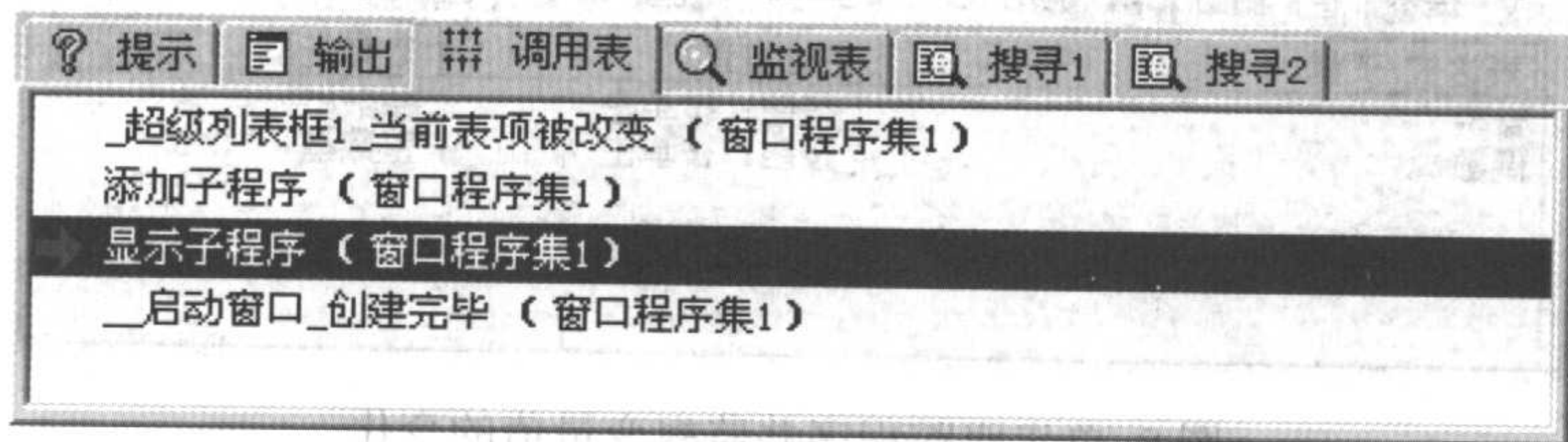


图 6 调用表中选择当前子程序

#### 4. 断点法

下面就结合前面介绍的“监视表”，使用程序断点来调试一个简单的程序。

在窗口中添加一个按钮，然后在按钮被单击的子程序中新建名为“奇数”和“偶数”的变量，然后使用记次数循环让这两个变量分别存放 100 以内的奇数和偶数。代码如下——

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
奇数	整数型			
偶数	整数型			

奇数 = -1

--> 计次循环首 (100, )

奇数 = 奇数 + 2

偶数 = 偶数 + 2

--- 计次循环尾 0





接下来，在子程序中的“记次循环首（）”命令行使用 F9 键设置断点。F5 运行程序后点击按钮，程序会暂停在设置断点的程序行，然后大家使用 Shift+F9 快捷键来调出查改变量对话框，将“奇数”和“偶数”变量分别加入到“监视表”中。如图 7 所示。

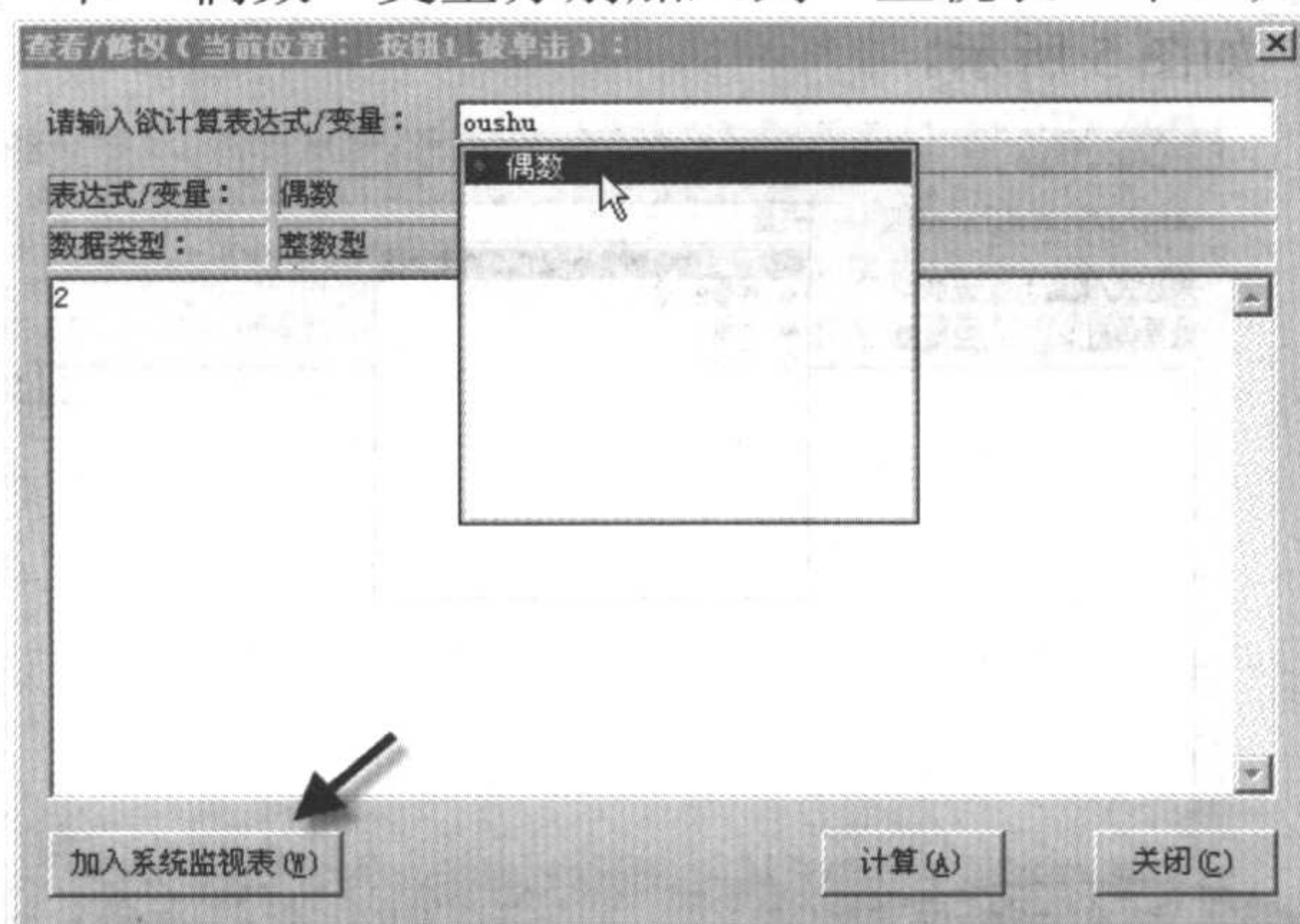


图 7 将“偶数”变量加入到监视表

加入到监视表后，将状态栏切换到“监视表”面板，然后使用 F8 键单步跟踪，连续点击 F8 键让程序单步运行，在运行过程中，就可以监视到 2 个变量中值的变化了。如图 8 所示。

表达式/变量	所处位置	数据类型	值
奇数	按钮1_被单击	整数型	9
偶数	按钮1_被单击	整数型	8

图 8 单步跟踪程序并监视变量值的变化

## 5. 设置条件断点

设置条件断点是易语言 4.0 版加入的新功能，可以在断点处进行逻辑判断，只有符合了条件断点设置的条件后，断点才会生效，这无疑提高了程序调试的效率。

设置条件断点的方法为，将光标移动到欲设置断点的命令行，然后执行菜单“运行”→“条件断点”，或者直接使用 Ctrl+F9 快捷键，执行后会弹出“条件断点设置对话框”，在该对话框中输入中断的条件表达式。如图 9 所示。

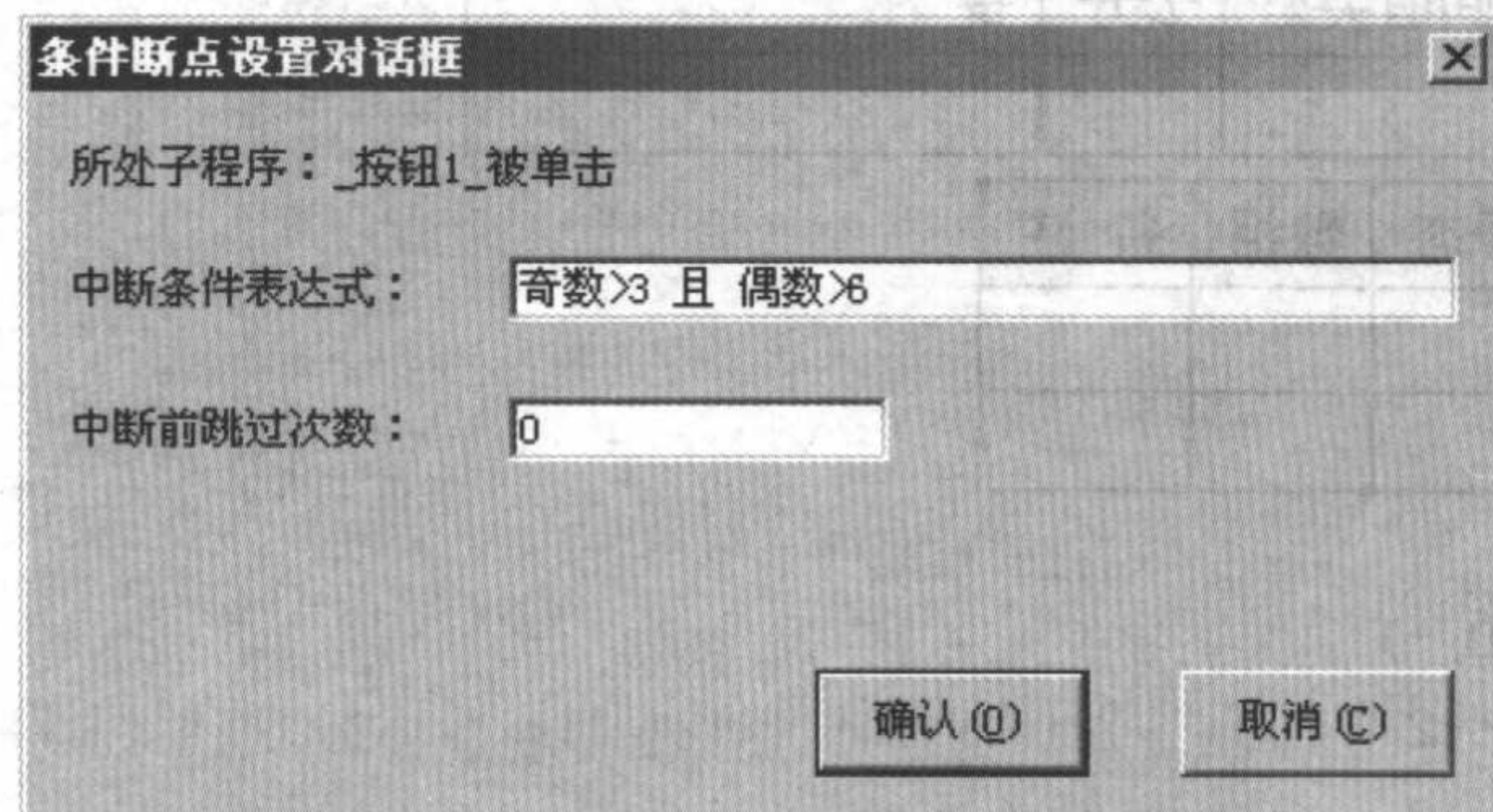


图 9 单步跟踪并监视变量值变化，多个条件用“且”或者“或”连接



从图 9 可以看出，条件表达式是可以使用“and”或“or”来连接多个条件的，另外，条件表达式还可以包括四则运算表达式和各级运算符，关于运算符，大家可以参看第二章中关于运算符的介绍。

“条件断点对话框”中还可以设置“中断前跳过次数”，该跳过次数表示，程序会在断点条件成立后跳过指定次数的断点，然后生效。例如，一个条件断点的条件成立了 12 次，当设置了中断前跳过次数为 2 时，则程序会在该断点条件第 3 次成立时，使断点生效，因为跳过了前 2 次生效了的条件断点。

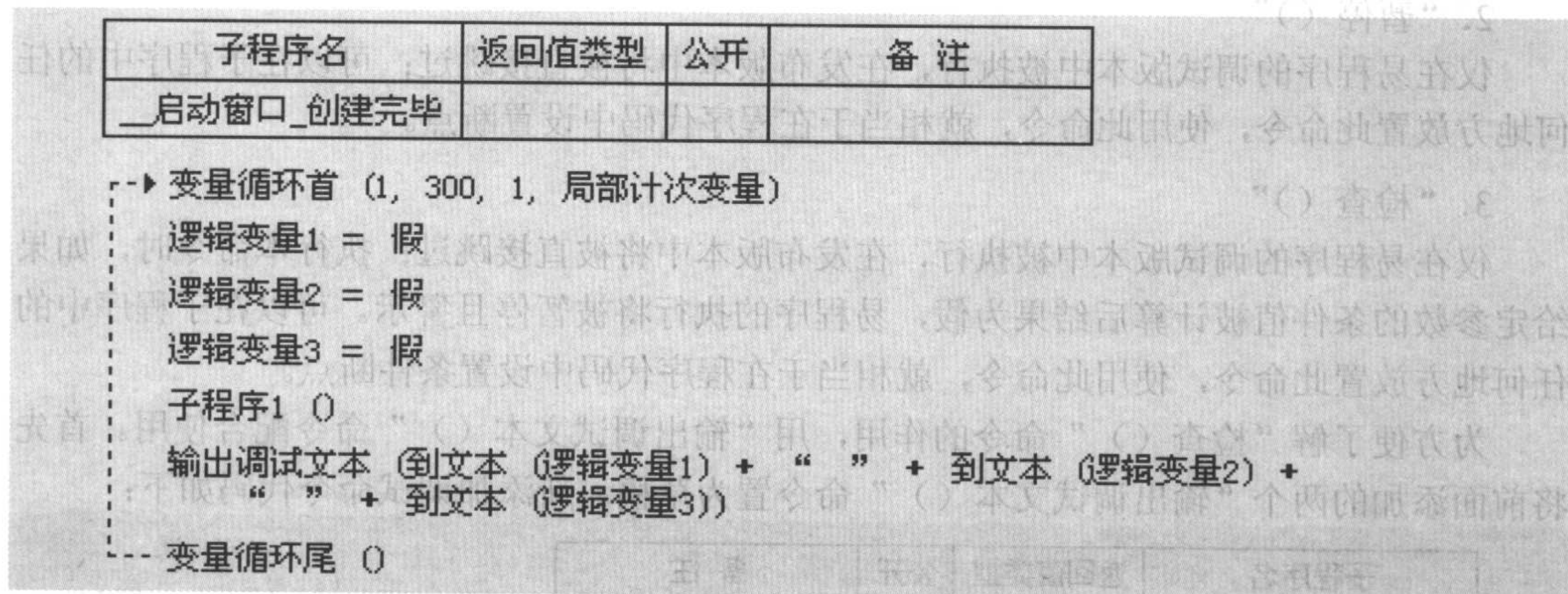
## 调试输出命令

以下被调试程序为“调试程序 2.e”。

### 1. “输出调试文本 ()”

仅在易程序的调试版本中被执行，在发布版本中将被直接跳过；使用本命令可以在易语言调试系统的“输出”面板中输出指定的文本行以帮助调试，该文本之前被自动加上一个星号(\*)，之后被自动加上回车换行符。

如果觉得“改写变量”面板只能显示当前被调试程序变量的一个值，无法与前面或后面程序运行时变量的值进行对比，十分的不方便。就可以用“输出调试文本 ()”命令实现这个功能。在“\_\_启动窗口\_创建完毕”事件子程序中添加“输出调试文本 ()”命令如下：



注意提供的参数数据类型必须为文本型。

按“F5”键运行程序，在“输出”面板中显示，如图 7 所示。

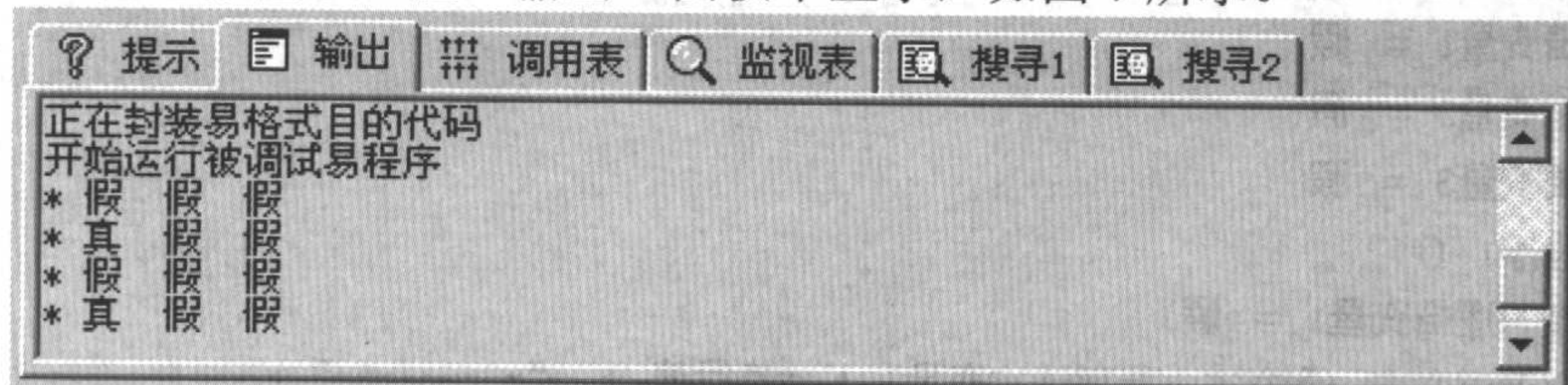


图 7 输出面板显示调试文本





如果调试时使用到多个“输出调试文本( )”命令，就必须提供被显示变量值的变量名或变量值的解释文本。在“子程序”中的最后一个如果真语句中添加“输出调试文本( )”命令如下：

```
--- 如果真 (局部计次变量 % 5 = 0)
    逻辑变量3 = 真
    输出调试文本 (“逻辑变量3: ” + 到文本 (逻辑变量3) + “ ” +
        “局部计次变量: ” + 到文本 (局部计次变量))
    编辑框1.内容 = 编辑框1.内容 + 到文本 (局部计次变量) + #换行符
```

显示结果如图 8 所示。

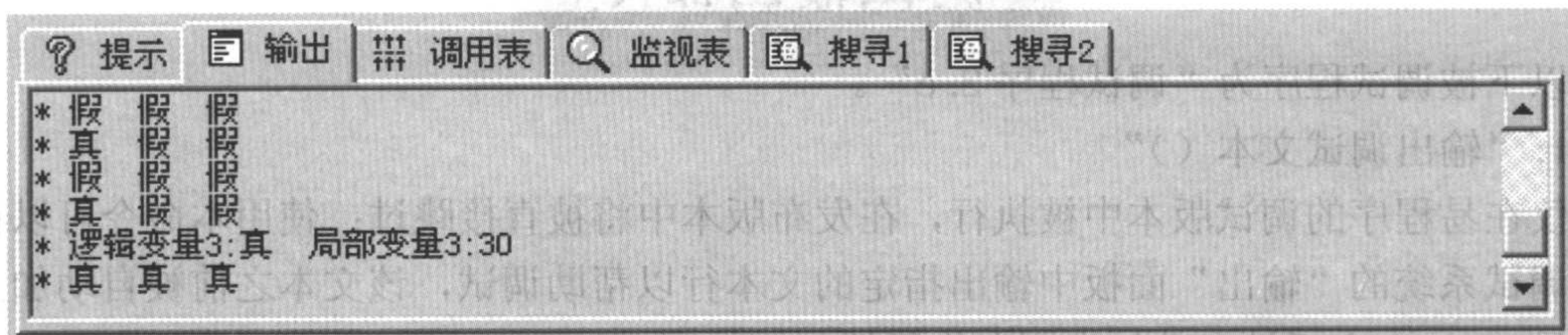


图 8 标记调试文本

可以很清楚地看到，只有“局部计次变量”的值为“30”时程序才会执行到最后一个如果真语句中，并使“逻辑变量 3”的值为真。

## 2. “暂停( )”

仅在易程序的调试版本中被执行，在发布版本中将被直接跳过；可以在子程序中的任何地方放置此命令，使用此命令，就相当于在程序代码中设置断点。

## 3. “检查( )”

仅在易程序的调试版本中被执行，在发布版本中将被直接跳过；执行本命令时，如果给定参数的条件值被计算后结果为假，易程序的执行将被暂停且警示。可以在子程序中的任何地方放置此命令，使用此命令，就相当于在程序代码中设置条件断点。

为方便了解“检查( )”命令的作用，用“输出调试文本( )”命令配合使用。首先将前面添加的两个“输出调试文本( )”命令置为草稿。新添加调试命令代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

```
--- 变量循环首 (1, 300, 1, 局部计次变量)
    逻辑变量1 = 假
    逻辑变量2 = 假
    逻辑变量3 = 假
    子程序1 ()
    检查 (逻辑变量1 = 假)
    输出调试文本 (到文本 (局部计次变量) + #换行符 + “-----”)
--- 变量循环尾 ()
```



按“F5”键运行程序，当“检查（）”命令中的条件不成立时，程序被暂停。提示如图9所示。

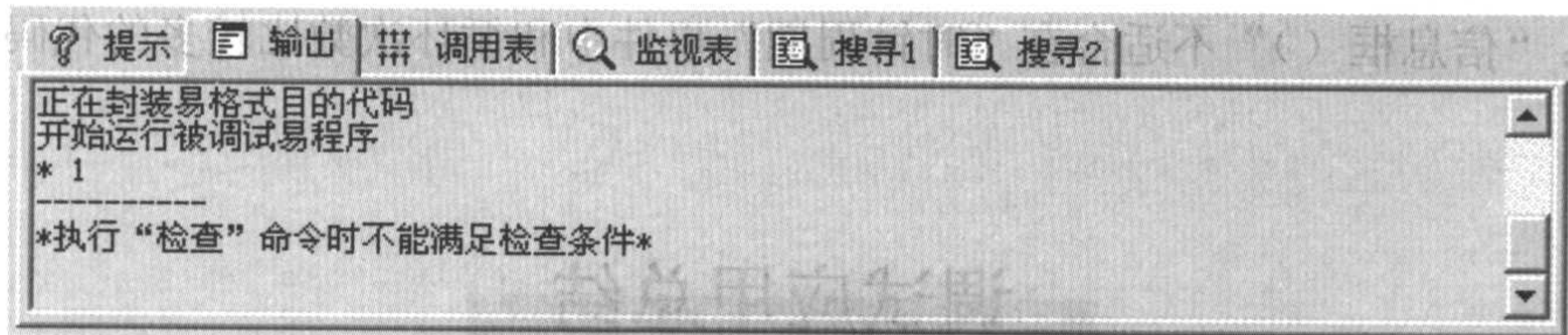


图9 “检查（）”命令被执行

从输出的文本可以看出，当“局部计次变量”为“1”时，程序无法进入“子程序1”中的第一个如果真条件语句中执行，所以“逻辑变量1”的值保持不变，为假，“检查（）”命令没有被执行。而“输出调试文本（）”命令正常输出文本（“局部计次变量”的值和“-----”）。当“局部计次变量”为“2”时，符合“子程序1”中第一个如果真条件语句，其中的代码被执行，“逻辑变量1”的值为真，“检查（）”命令的条件不成立，“检查（）”命令被执行并输出提示。

#### 4. “是否为调试版（）”

如果当前 EXE 易程序执行文件为易语言编辑环境调试运行程序时编译出来的调试版本，返回真。否则表明为发布版本，返回假。

#### 5. “信息框（）”

在对话框中显示信息，等待用户单击按钮，并返回一个整数告诉用户单击哪一个按钮。

“信息框（）”是软件中必不可少的组成部分，是用户与软件之间的一种通信通道。但在程序的调试过程中也被经常使用到，而且可以达到多种调试方法的功能。如：断点、暂停（）、输出调试文本（）等方法。将程序中的调试命令全部置为草稿，添加“信息框（）”，“子程序1”代码如下：

```

--- 如果真 (局部计次变量 % 2 = 0)
    逻辑变量1 = 真
    --- 如果真 (局部计次变量 % 3 = 0)
        逻辑变量2 = 真
        --- 如果真 (局部计次变量 % 5 = 0)
            逻辑变量3 = 真
            信息框 (局部计次变量, 0, )
            编辑框1.内容 = 编辑框1.内容 + 到文本 (局部计次变量) + #换行符

```

按“F5”运行程序，信息框提示“局部计次变量”为“30”，信息框被第一次执行，程序被暂停。如图10所示。



图10 信息框调试





使用“信息框( )”显示数据不需要考虑是文本、数值、逻辑值或日期时间。如果同时显示不同类型变量和数据值,必须先转换到文本型。

注意:“信息框( )”不适合在“时钟周期”事件中和循环次数比较多的代码中使用。

### 调试应用总结

程序中出现的错误通常分为“语法错误”和“逻辑错误”。

所谓“语法错误”是指程序代码不符合易语言语法。这种错误最容易发现和修改:首先在代码输入的时候,系统就会检查并发现一部分语法错误;其次在程序运行的时候,系统执行到有语法错误的代码行,也会发现并指出错误原因。由此可见,系统会帮助找出所有的语法错误,只要按照其提示信息进行相应的修改即可。

所谓“逻辑错误”是指程序流程上、处理上的错误。含有“逻辑错误”的程序能够正常执行,只是执行结果不正确。这类错误系统是不可能发现的,只有靠编程者自己去寻找。

实际应用中,通常是将“断点”“单步跟踪”“查改变量”(以及调试输出)等调试方式结合起来使用。

一般的程序调试步骤是这样的:

1. 运行程序,执行所有的程序功能,从而找出并修改所有“语法错误”;
2. 通过分析判断,找到可能有“逻辑错误”的代码段;
3. 在“有逻辑错误的代码段”前面设置“断点”;
4. 运行程序,待程序在“断点”处中断后,使用“单步跟踪[F8]”、“单步跟踪进入[Shift+F8]”、“跟踪返回[Ctrl+Shift+F8]”、“执行到光标处[Ctrl+F8]”等调试命令跟踪程序的运行。跟踪过程中,随时观察各变量值的变化(通过状态夹中的“查改变量”,可看到所有全局和局部变量),必要时使用调试输出命令将变量值输出。通过跟踪,一般能发现程序出错的原因。
5. 终止程序运行,修改代码,继续调试。



## 附录二 易语言编译与发布

易语言程序编写完成之后，如果想要该程序脱离易语言编程环境独立运行，就需要对其进行编译。在易语言中，程序编译有三种方法，分别是“编译”、“独立编译”、“编译生成安装软件”。其中“编译”是仅对易程序代码进行编译；“独立编译”会将易程序代码与所需支持库一同编译，生成可在非安装易语言开发环境的操作系统中独立运行的程序；“编译生成安装软件”会在独立编译的基础上增加安装向导功能。

注意：编译程序前可以在系统配置对话框（易语言系统菜单的“程序”子菜单的“系统配置”菜单项）设置编译选项。如图 1 所示。

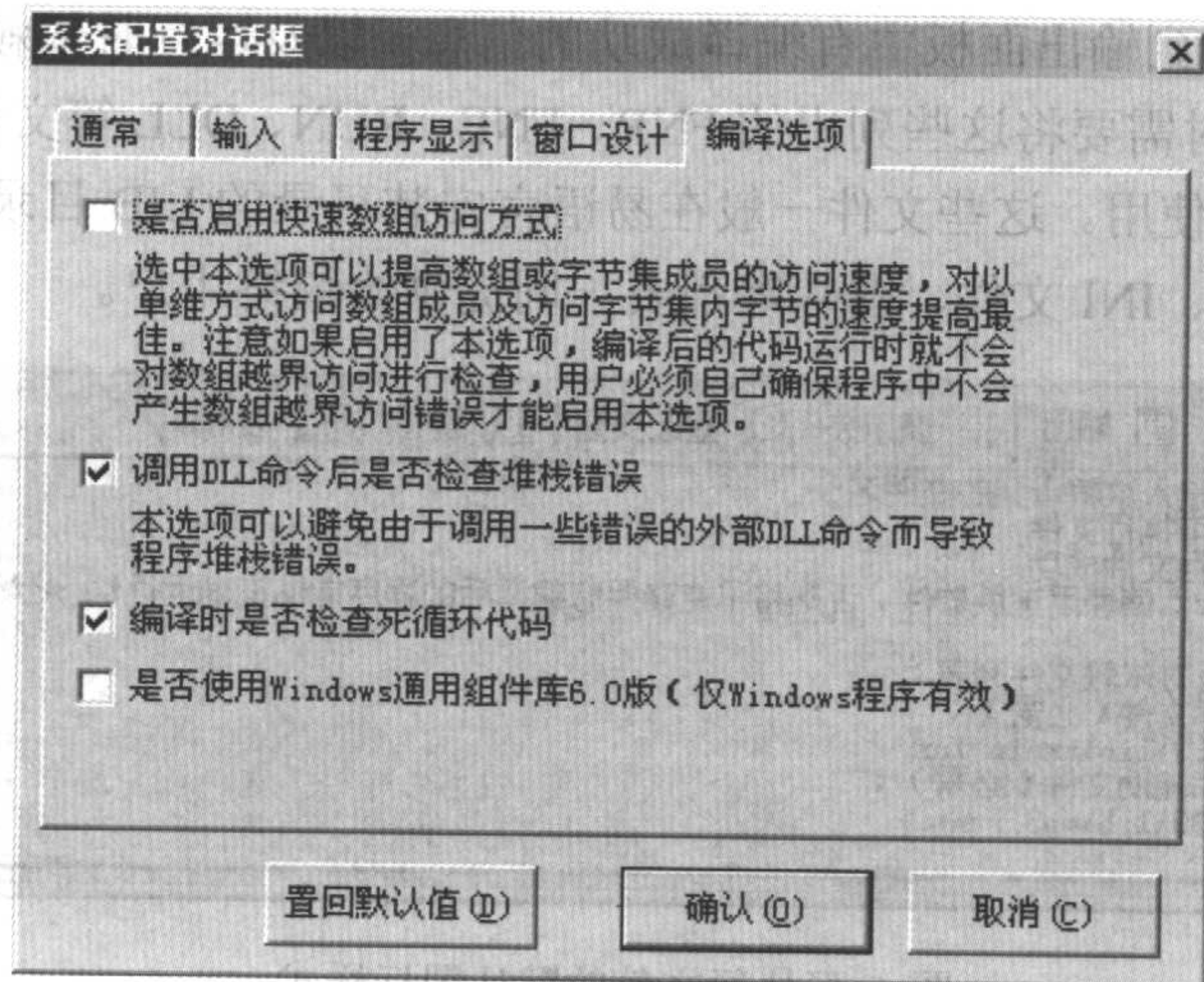


图 1 系统配置对话框

以下讲解全部假定以“Windows 窗口程序”为对象进行编译，特此说明。

### 非独立编译

非独立编译是只对程序源码部分而不包含易语言支持库的编译。非独立编译后的程序必须有程序中所使用到的支持库文件才可以运行。所以非独立编译后的程序如果要发布，必须将程序使用到的支持库文件（易语言支持库文件都存放在易语言安装目录中的“lib”目录中）和非独立编译后的文件一同发布。

1. 选择菜单“编译”→“编译”选项。如图 2 所示。



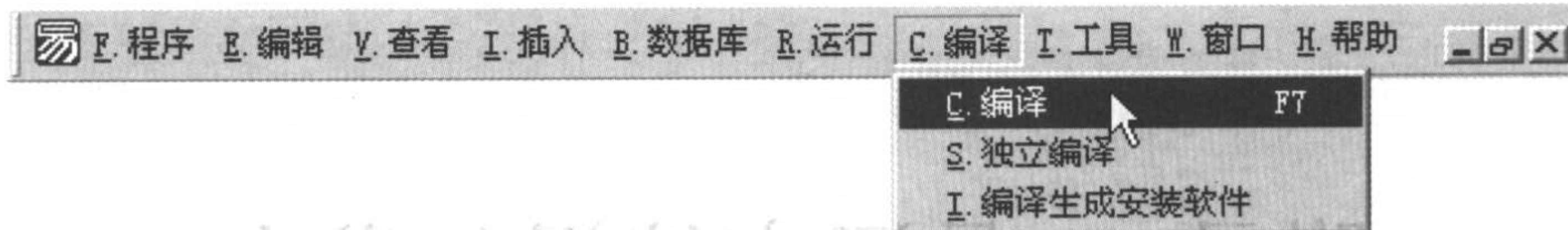


图 2 程序非独立编译

2. 在弹出的对话框中，选择所要保存文件路径，保存为 EXE 文件。编译后的可执行文件的默认图标样式如图 3 所示。



图 3 可执行文件的默认图标样式

3. 编译后，会看到输出面板中有编译成功的信息，以及运行时依赖的文件列表，如图 4 所示。大家在发布时需要将这些列出的 FNR、FNE、RUN、DLL 等文件与上述 EXE 文件一同提供给最终用户使用。这些文件一般在易语言安装目录的 LIB 目录下找到。如果有其他用到的数据库文件、INI 文件、图片等都要一同提供给最终用户。

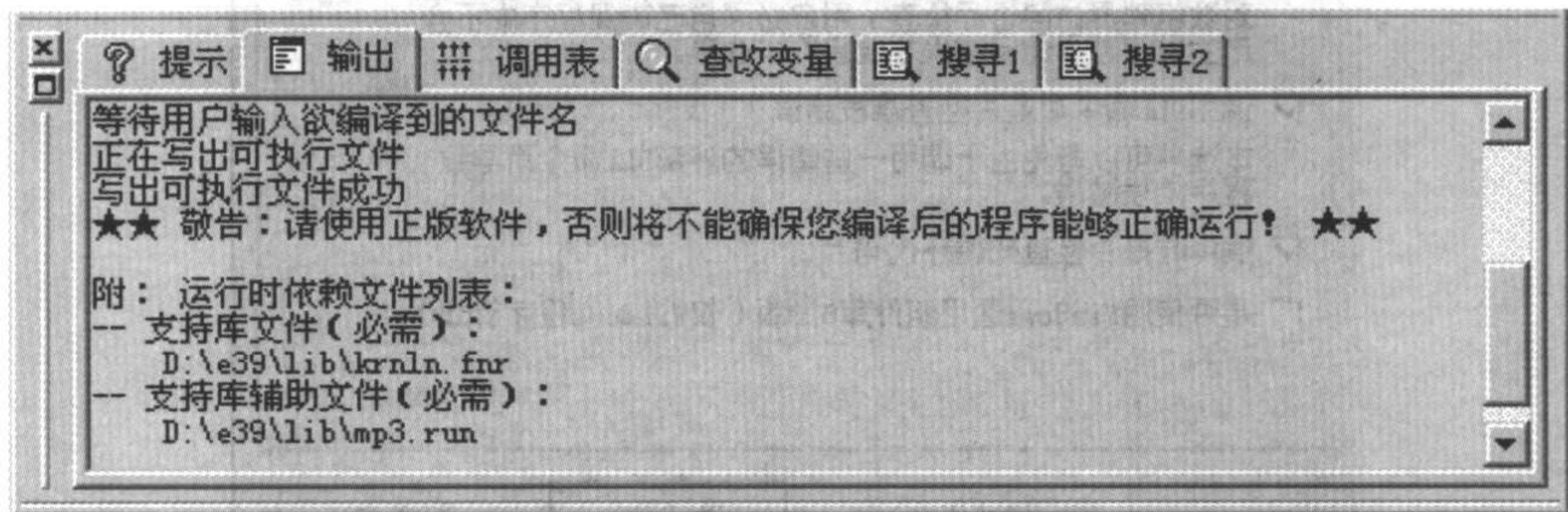


图 4 可执行文件的默认图标样式

## 独立编译

易语言的“独立编译”可把易程序独立运行所需要的文件全部封装到可执行文件当中，使其可以完全独立的运行。程序中所用到的支持库会自动全部封装进去，而其他外部文件，如 DLL、OCX 等，可以在编译时，由开发人员自由选择是否封装到可执行文件之中。

1. 选择菜单“编译”→“独立编译”选项。
2. 在弹出的窗口中显示了将要保存到 EXE 文件中的所有文件，可以选择一项或多项



进行编译，而灰色项目为必需的。如图 5 所示。

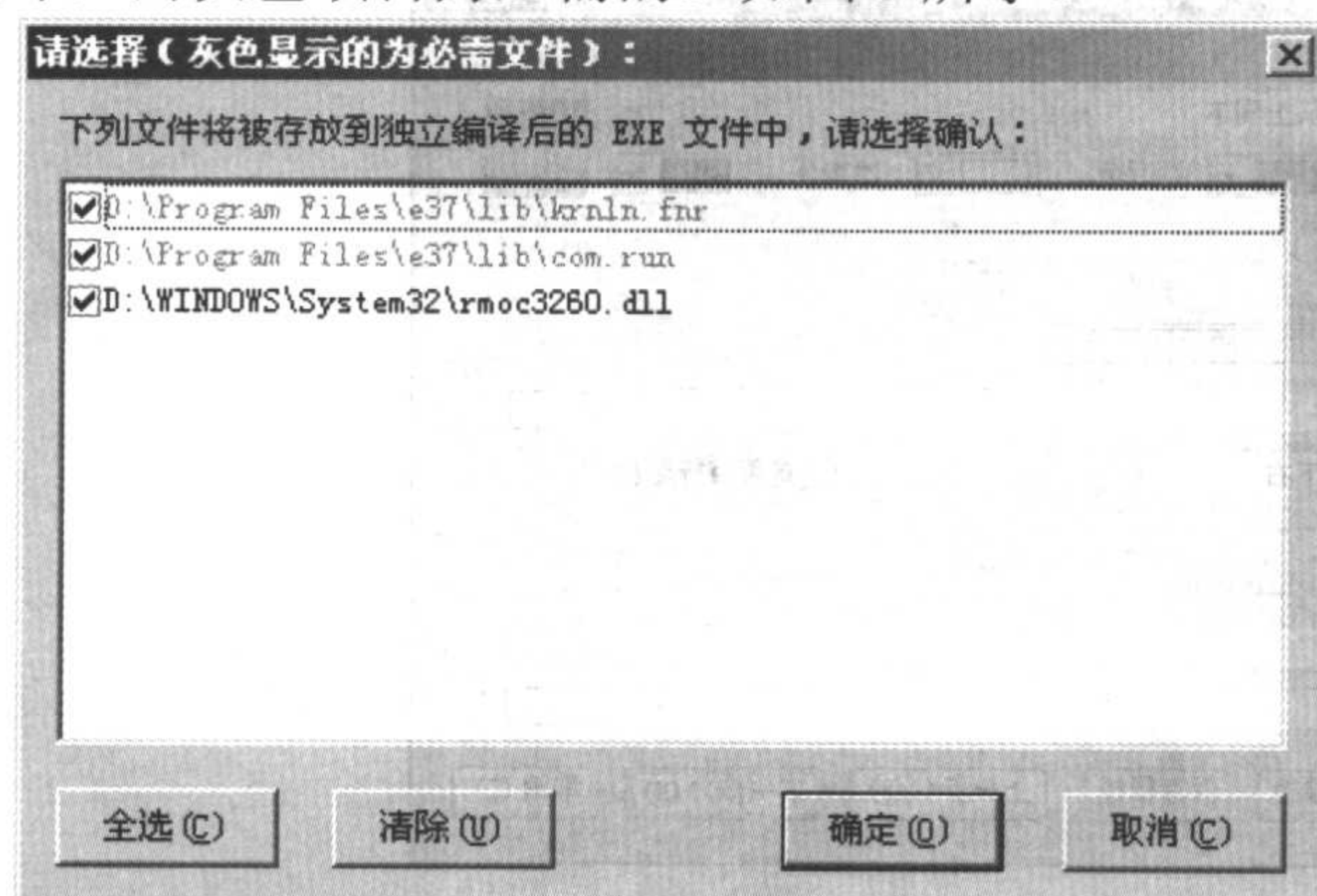


图 5 选择独立编译项目



图 6 可执行文件的图标样式

3. 在弹出的对话框中，选择所要保存文件路径，然后保存为 EXE 文件。  
独立编译后的可执行文件的默认图标样式如图 6 所示。

## 程序发布

为方便程序的分发，易语言提供一个标准安装向导程序，通过友善的向导界面提示终端用户安装易程序，同时也避免在程序发布时遗漏重要的文件。

1. 选择“程序”菜单中的“编译生成安装软件”命令；

注意：如果要发布的程序没有保存，系统会自动弹出一个信息框，提示保存。

2. 确定即将生成的安装包文件的保存路径和文件名称，点击“保存”按钮继续；

3. 添加安装软件时的窗口标题以及软件作者的声明和许可协议，点击“下一步”按钮继续。如图 7 所示。

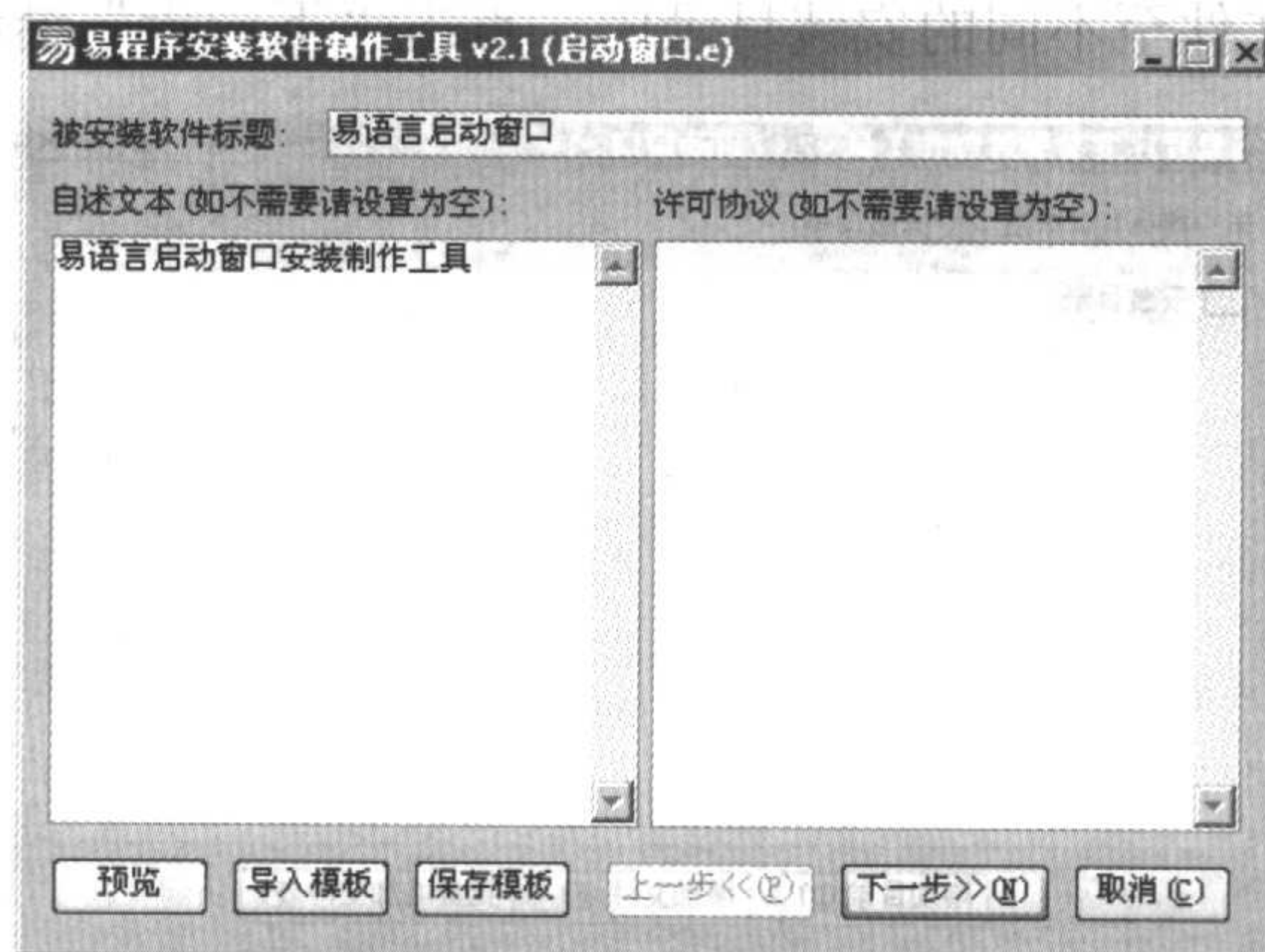


图 7 添加发布信息

4. 详细设置安装窗口的外观，点击“下一步”继续。如图 8 所示。



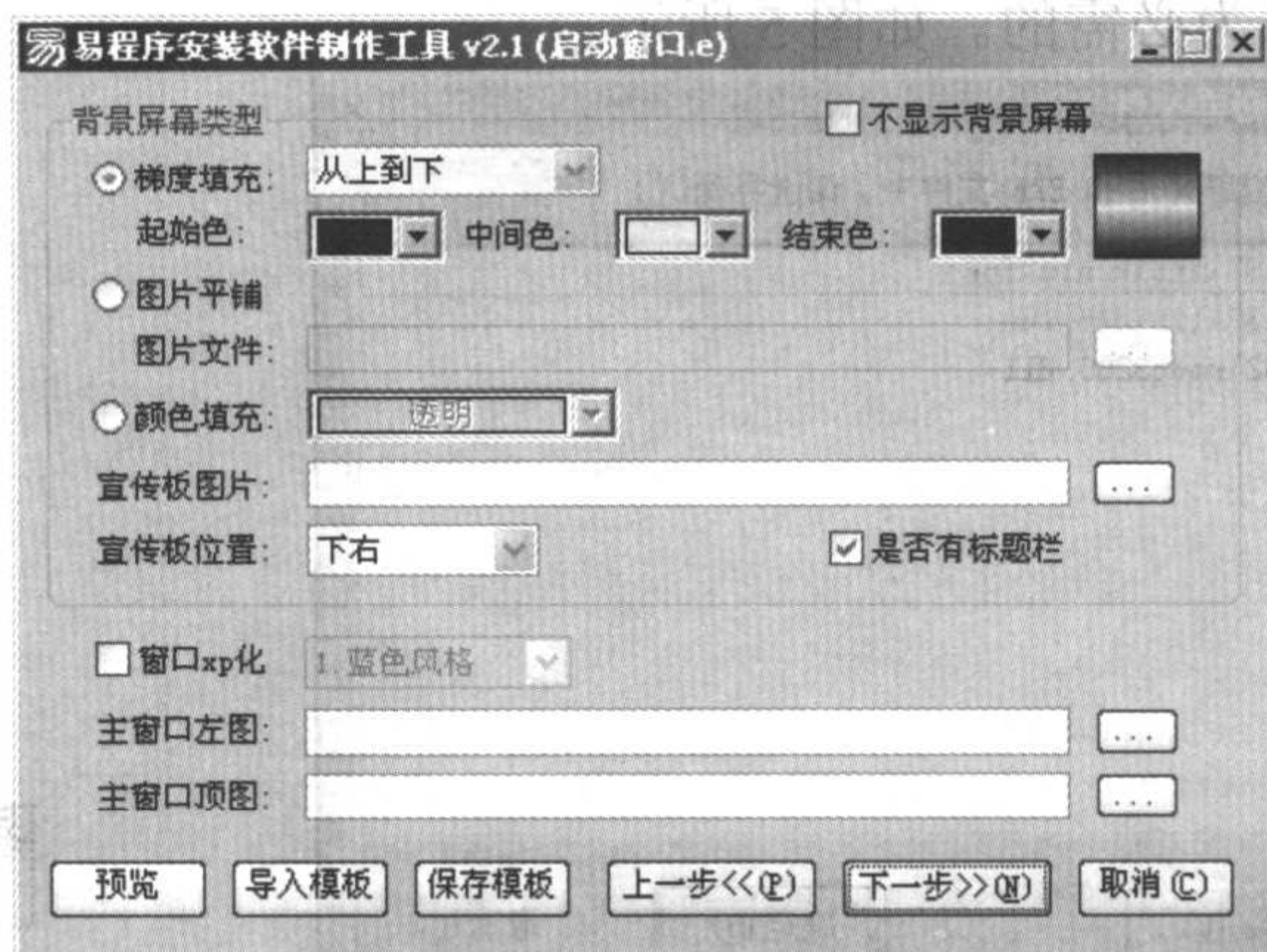


图 8 设置安装软件时的窗口外观

5. 选择需要加入到安装文件的基本文件，灰色为必选项，点击“下一步”继续。如图 9 所示。

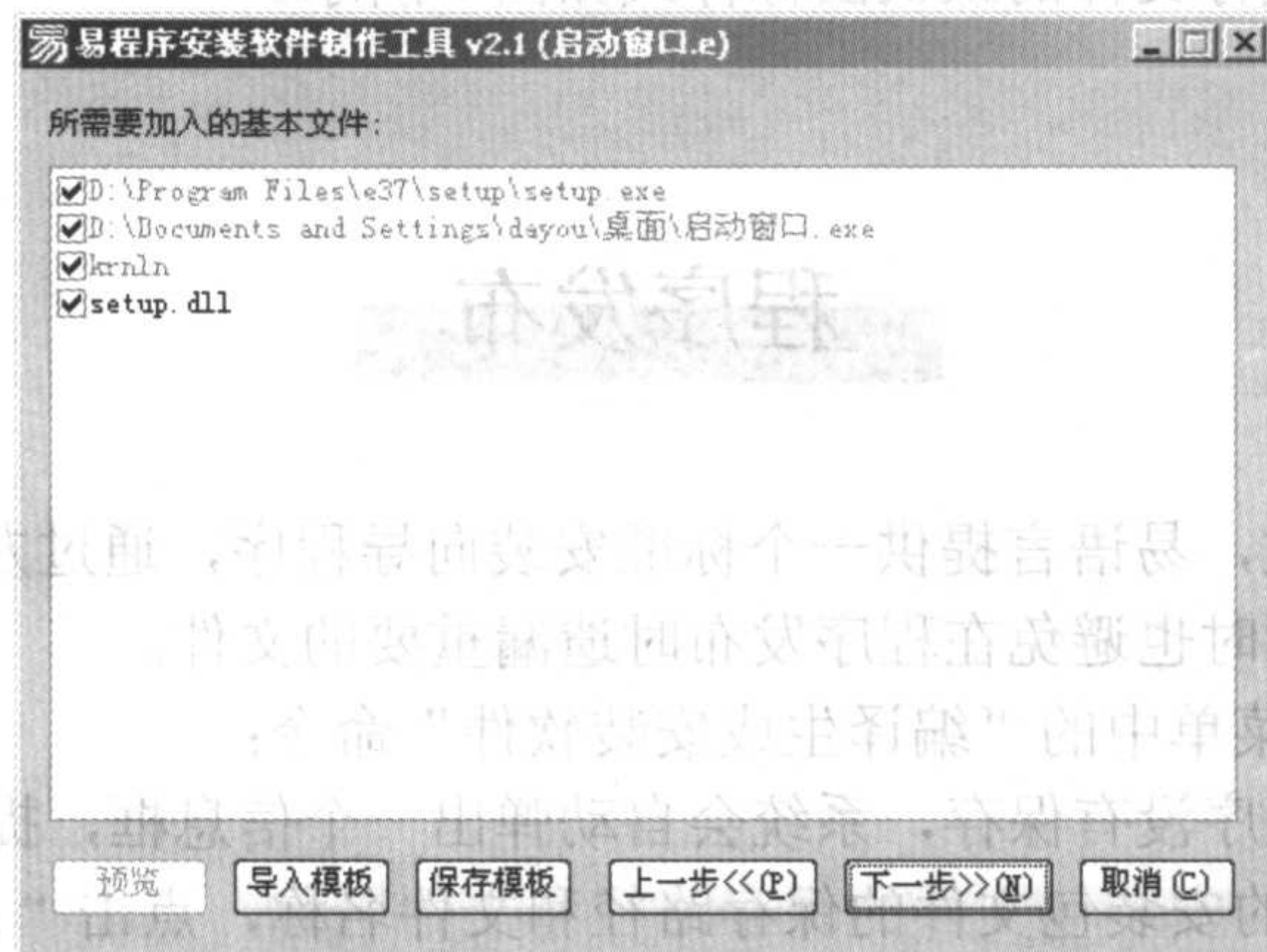


图 9 选择安装文件的基本文件

6. 添加其他相关文件至不同的安装目录中，点击“下一步”继续。如图 10 所示。

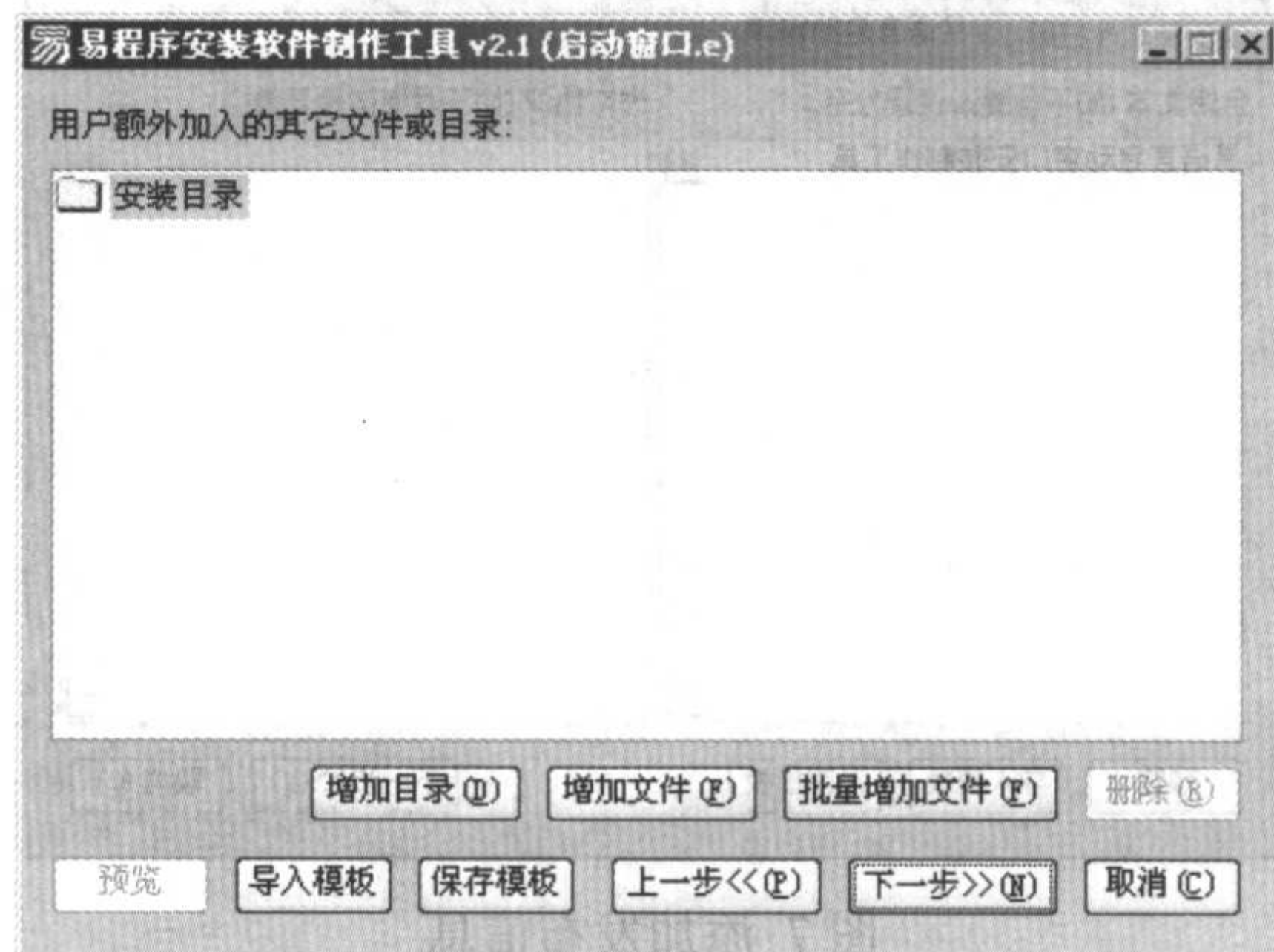


图 10 分类相关文件到不同的文件夹中



7. 设置软件默认的安装目录及相关选项, 点击“下一步”继续。如图 11 所示。

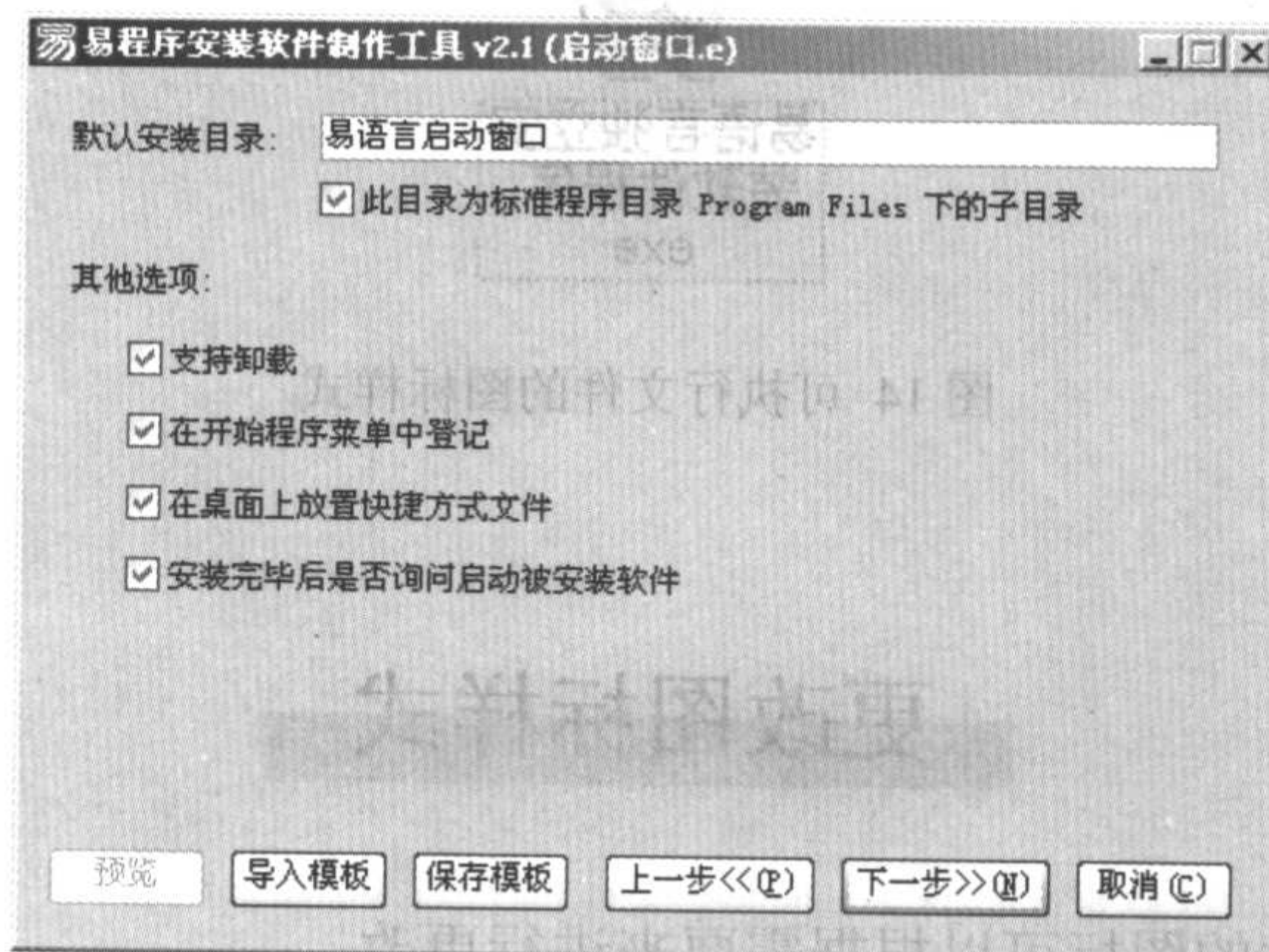


图 11 设置软件安装时的默认安装目录

8. 确定安装包文件属性, 点击“确定”按钮。如图 12 所示。

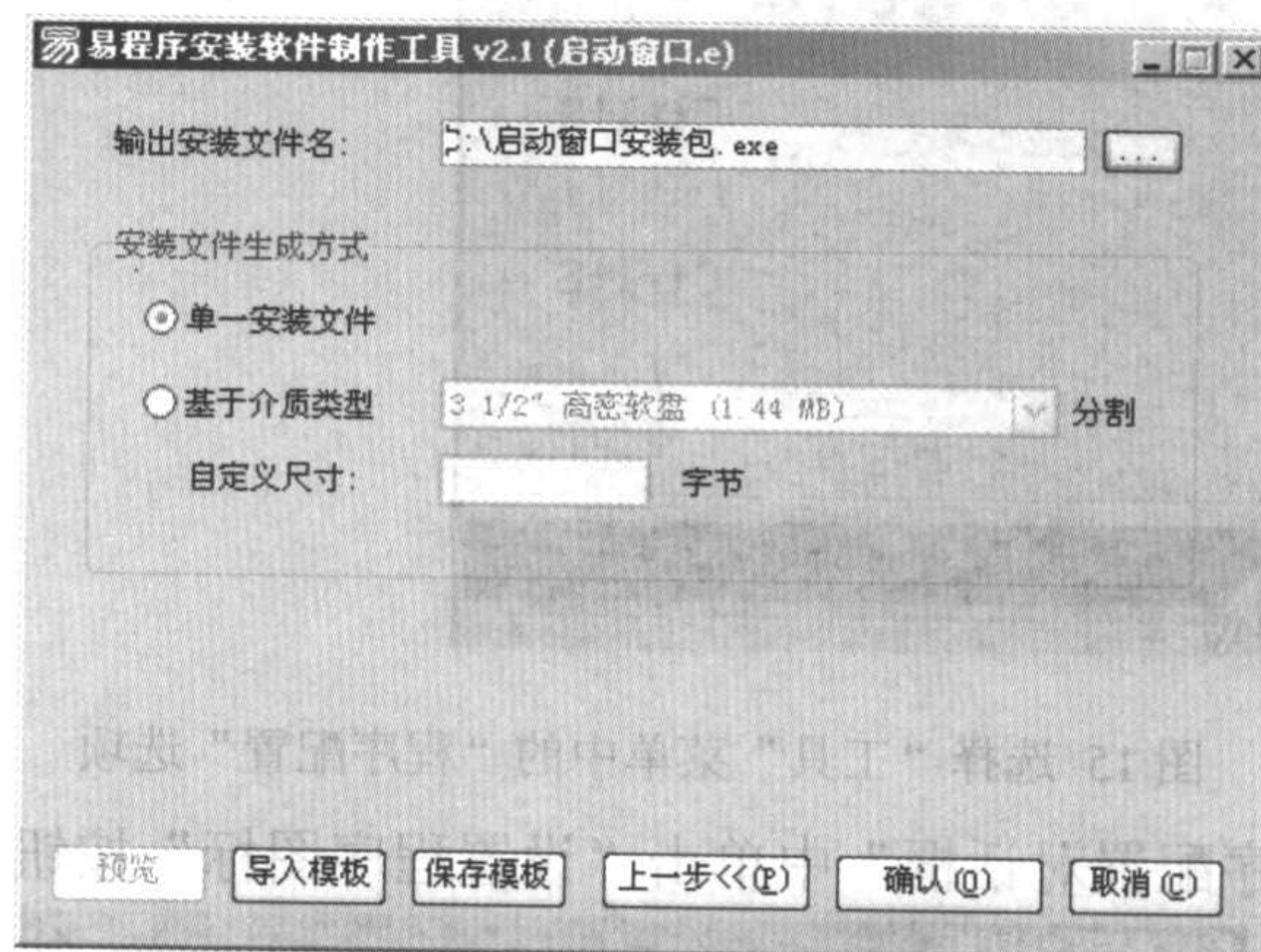


图 12 选择保存制作后的安装文件的目录

9. 编译打包工作完成之后, 会弹出一个“生成安装软件成功”的提示窗口。如图 13 所示。

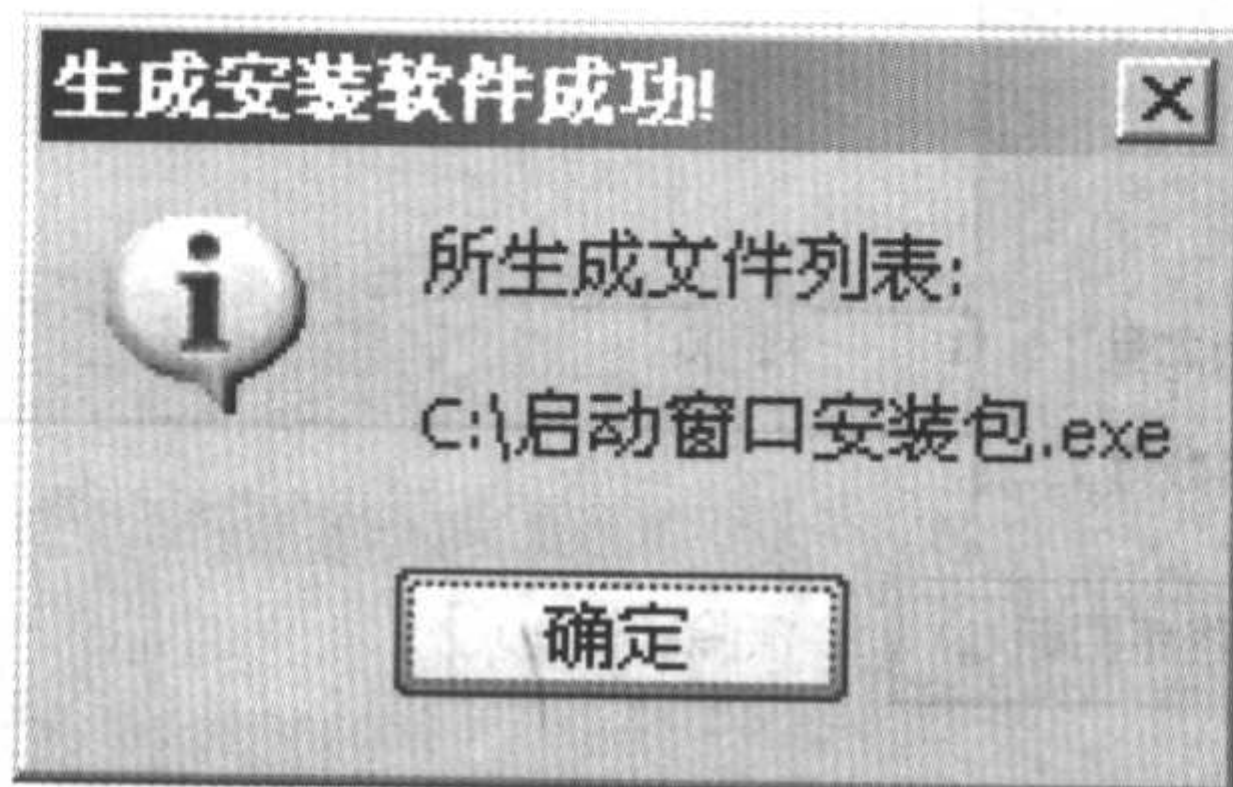


图 13 提示生成安装软件成功

安装向导编译所生成的可执行文件默认图标样式如图 14 所示。





易语言独立安  
装软件程序.  
exe

图 14 可执行文件的图标样式

## 更改图标样式

易语言可执行文件的图标可以根据需要进行更改。

1. 选择菜单“程序”→“配置”选项。如图 15 所示。

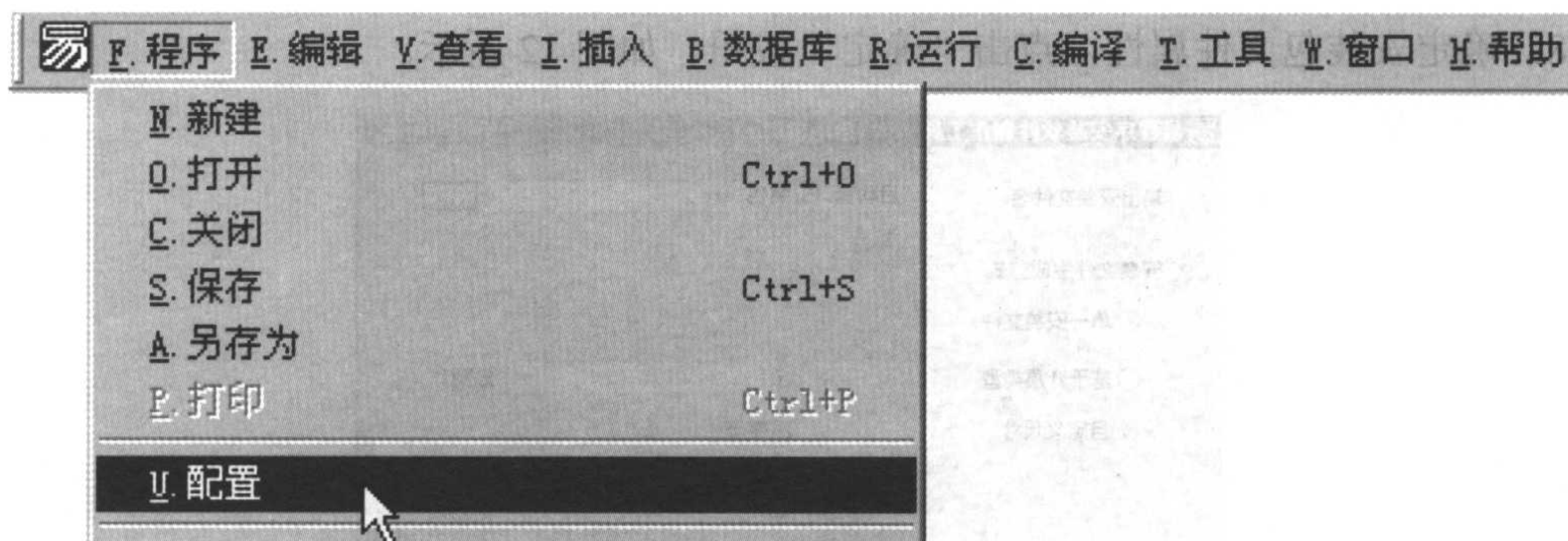


图 15 选择“工具”菜单中的“程序配置”选项

2. 在弹出的“程序配置对话框”中单击“设置程序图标”按钮。如图 16 所示。

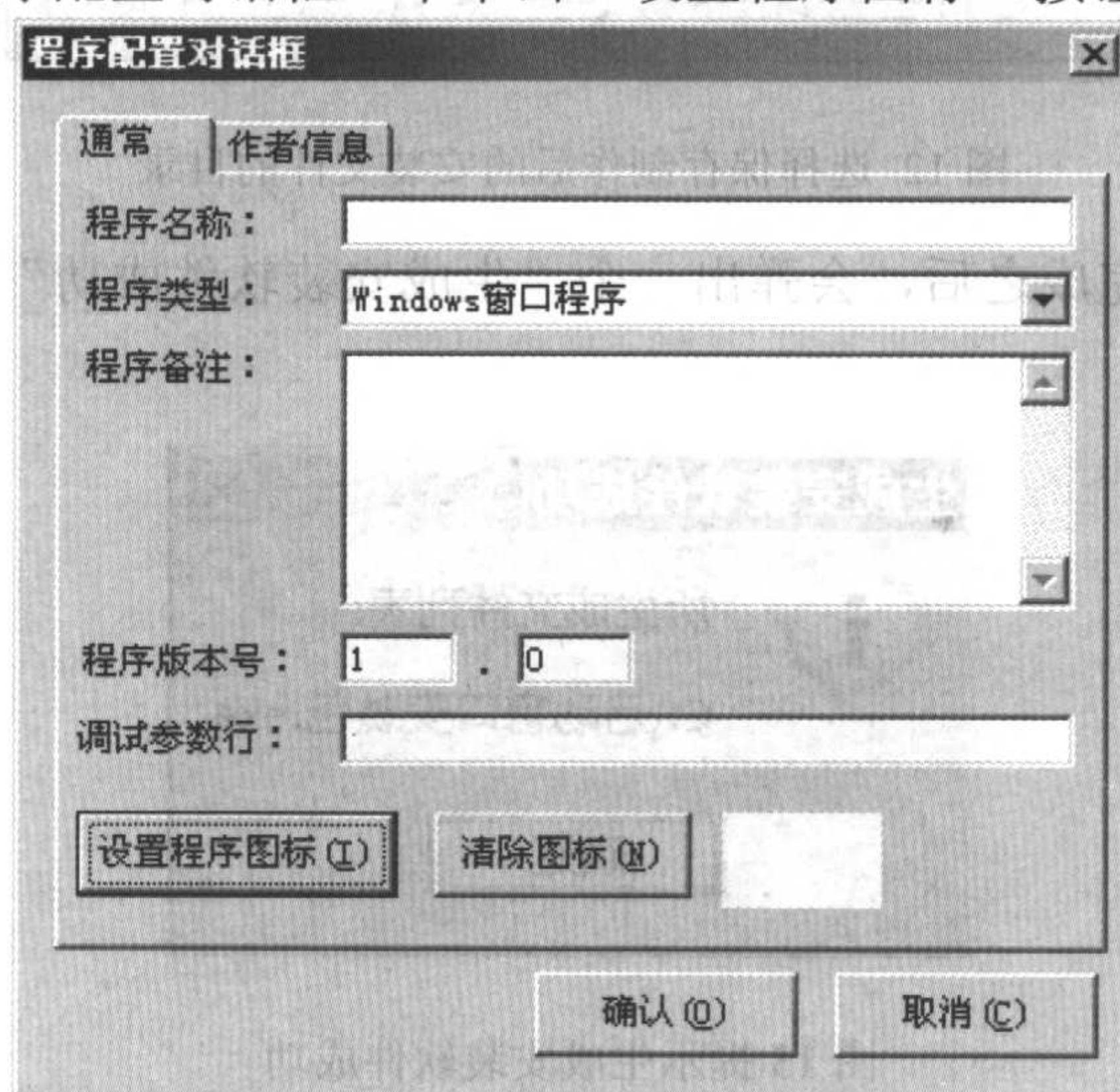


图 16 程序配置对话框



3. 在弹出的“请输入图标文件名:”对话框中,选择所要更改的图标样式的“.ico”文件图形。如图 17 所示。

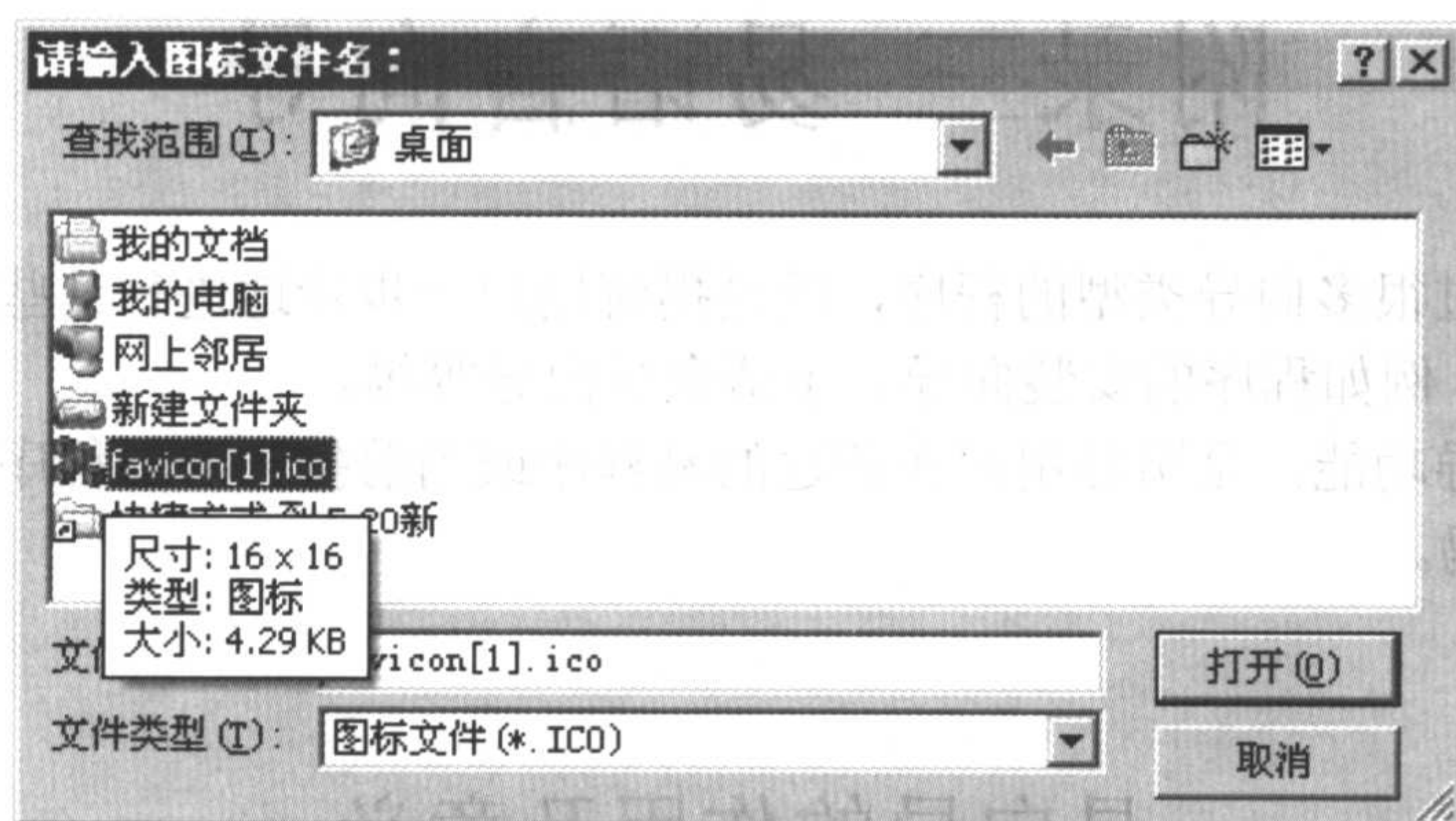


图 17 选择更改图标

4. 在所弹出的提示对话框中,单击“是”按钮,确认要设置该图标文件为程序图标。如图 18 所示。

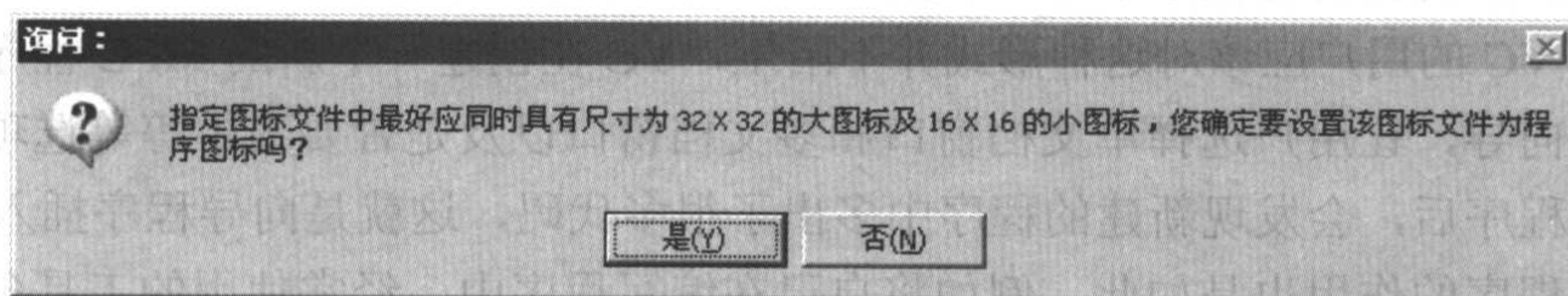


图 18 提示对话框

5. 程序配置对话框中,单击确认,结束更改图标的所有操作。

当然,大家也可以通过对窗口的“图标”属性分别操作设置 ICO 图片文件,以实现分别为窗口设置不同的图标。

图 18 中所要求的图标类型有 32×32 的大图标和 16×16 的小图标的 ICO 文件可由专业制作 ICO 的工具生成,这些工具网上非常多,大家可以下载使用。也可以在网上直接找到现成的 ICO 图标。





## 附录三 易语言向导

相信大家用过很多向导类型的程序，就是提醒用户一步步地进行某些操作，最后得到用户需要的结果。例如程序的安装向导、申请会员向导等等。

易语言的向导功能，是引导用户在新建的易程序或当前打开的易程序中添加或插入自己需要的一些代码。

### 易向导的作用及意义

简单来说，易向导的功能就是提供一段完整的源程序代码，然后使用一个向导程序来引导用户一步步的在自己程序中导入或插入已经写好的源程序代码。

使用过 VC 的用户应该对这种形式并不陌生，VC 在创建一个新的 MFC 程序后，会弹出应用程序向导，让用户选择单文档窗口和多文档窗口以及是否要菜单等等选项，用户选择并生成新程序后，会发现新建的程序中多出了很多代码，这就是向导程序插入的代码。

易向导程序的作用也是如此，例如将自己在编写程序中，经常使用的工具条、菜单项和代码等，编写成易向导，然后每次新建易程序时，就插入这些组件和代码。

制作向导的用户需要先写一个完整的源程序（实现全部需要实现的功能），该源程序就是一个模板程序（最后生成的代码就是通过对该模板进行编辑后的结果），然后编写一个向导界面，提供给别人一些选项，例如是否要工具条、是否要菜单等等，最后根据最终的选项，对已经编写好的模板程序进行删除和编辑，最后插入到使用向导人的程序中一段编辑后的代码。

易向导支持库中提供的命令就是用来对源程序（源代码）进行删除和编辑的。

使用易向导，可以方便代码的相互交流，以及方便他人根据需要使用自己的代码；还可以对自己经常使用的一些重复的代码编写成向导程序，当需要使用这些代码时，就使用向导程序将常用的代码插入到自己的程序中。

### 易向导的使用方法

上面介绍了易向导程序的概念，大家知道易向导程序的用途了，下面就使用一个制作好的向导程序来介绍一下向导程序的使用方法。

首先，大家要知道，易向导程序是一个非独立编译的易程序，该程序存放在指定的文



文件夹中（易语言安装目录下的“wizard”文件夹）。

当易语言新建对话框打开后，对话框的树型框中可以找到“通用向导创建”项目，双击该项目下的“常用向导”子项目，所有的向导程序就会显示在对话框中，用户在新建对话框中调用向导程序后，向导程序会将代码插入到新建的易程序中。如图 1 所示。

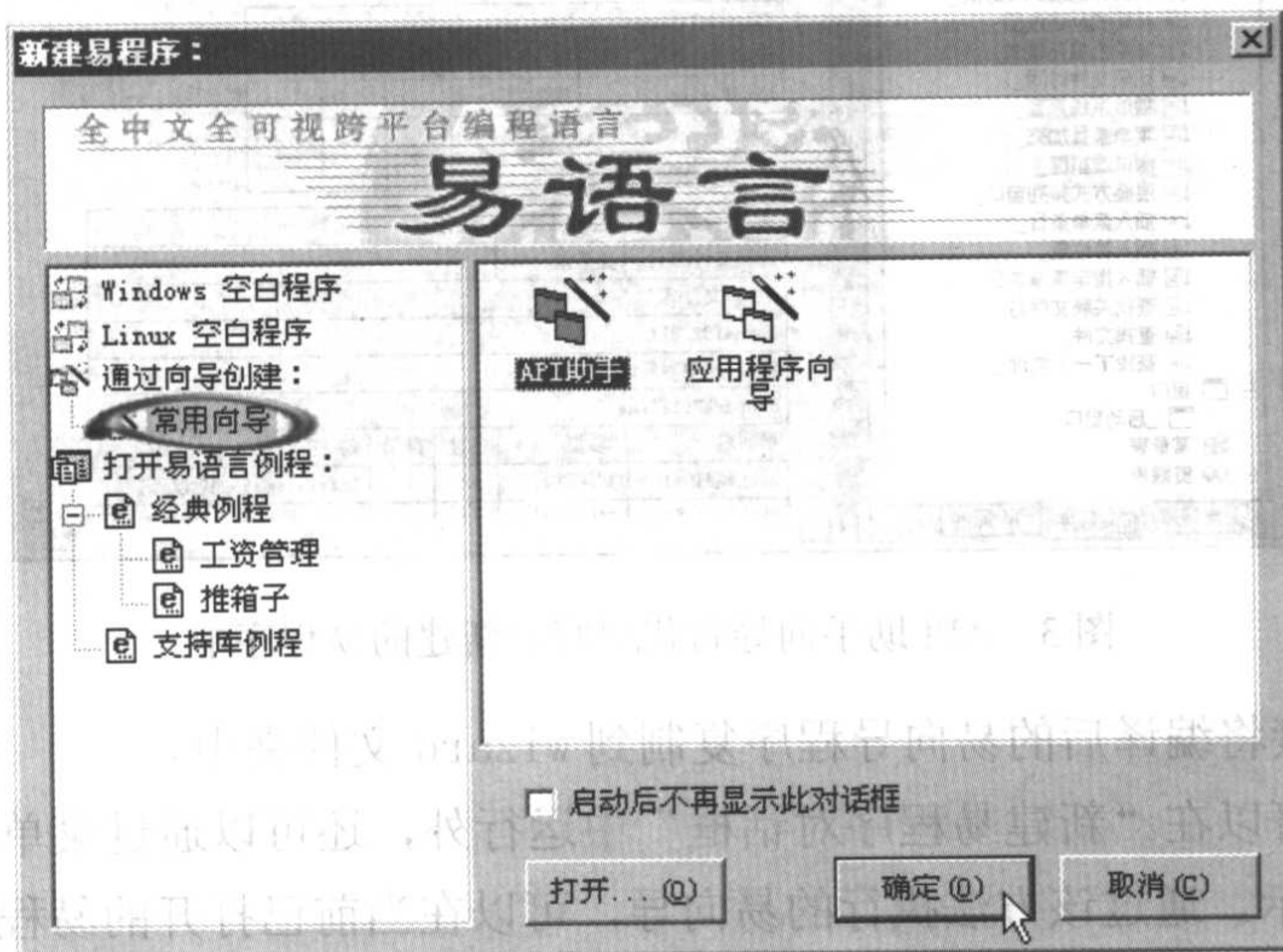


图 1 新建易程序对话框中使用易向导

调用 API 助手向导后，弹出向导程序的主界面，如图 2 所示。

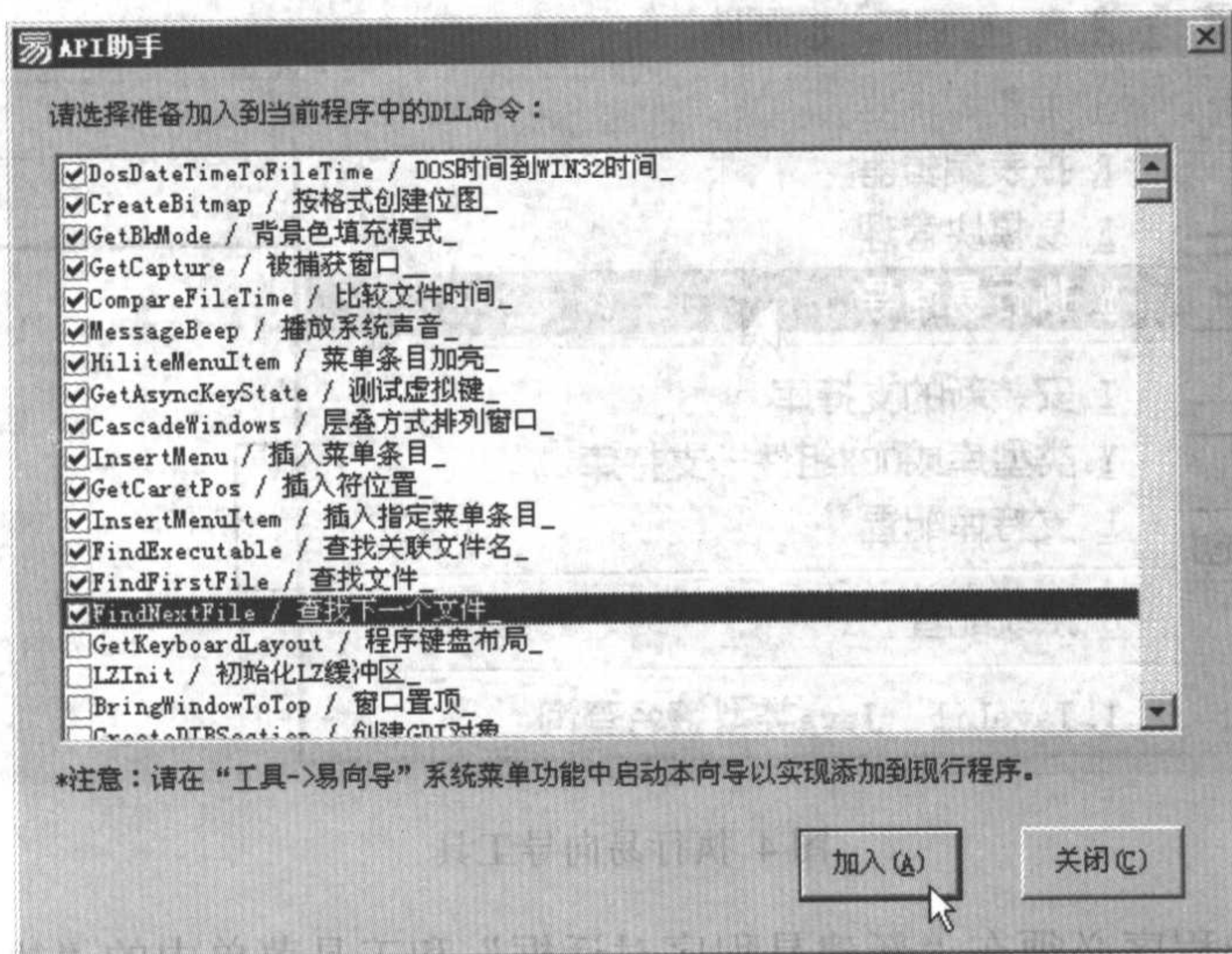


图 2 API 助手向导界面

在窗口中，选中一些 API 命令，然后点击加入按钮，API 助手向导会将选中的 API 写进新建易程序的 DLL 命令中，如图 3 所示。



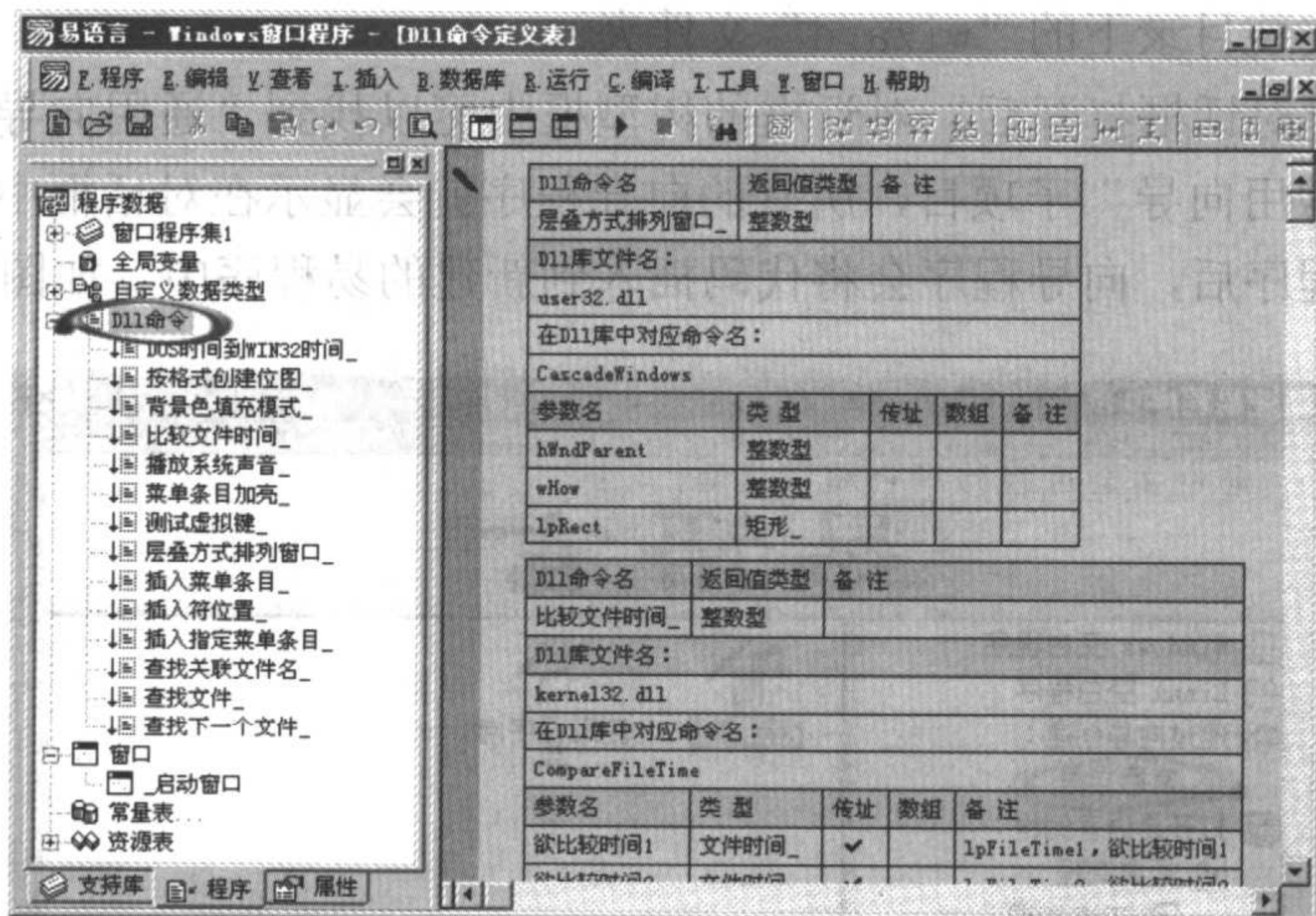


图3 API助手向导将代码写到新建的易程序中

用户可以直接将编译后的易向导程序复制到 wizard 文件夹中。

易向导除了可以在“新建易程序对话框”中运行外，还可以通过菜单：工具→执行易向导，如图4所示。通过该方法执行的易向导，可以在当前已打开的易程序中插入代码。这里要注意，如果是编写可以在已打开的易程序中插入代码的向导，在写易向导代码时，还要对“写出程序（）”命令的参数进行设置，关于易向导的编写方法后面会详细介绍。

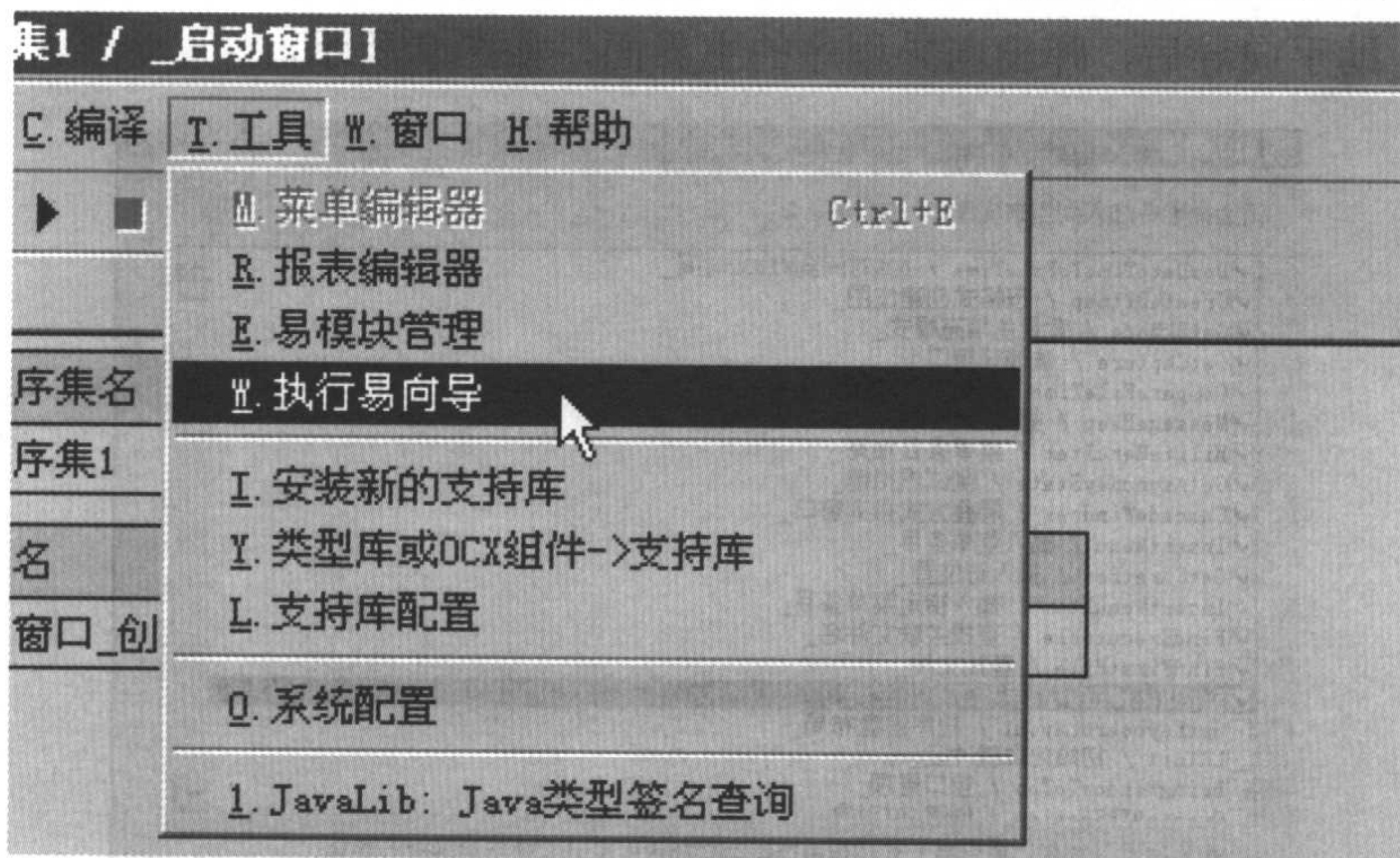


图4 执行易向导工具

所以，易向导程序必须在“新建易程序对话框”和工具菜单中的“执行易向导”中进行调用，任何易向导程序脱离易语言单独运行，都是没有效果的。这是因为向导程序要和易语言程序要进行通讯，易语言需要给易向导程序提供写出代码的接口。

注意：想要测试向导程序写出程序是否成功，也要将易向导程序编译后进行调试。



## 易向导的编写

易语言的向导制作需要使用易向导支持库提供的命令，该支持库中的命令都是针对模板程序中的代码和组件（即源代码程序）进行操作，例如删除、复制、修改子程序或组件等操作，从而生成用户需要的代码。

下面就介绍一下易向导支持库的使用方法和部分重要命令：

易向导程序一般分为两个步骤，首先是用户的设置，用户根据向导程序提供的界面，对需要生成的功能代码进行设置，这些设置，可以使用变量保存起来；其次，就是向导程序根据用户的选择，对模板程序进行修改然后写出代码。

在这个过程中需要注意的是：对模板程序的修改是在写出代码前一次性完成的，而并不是在用户选择时进行修改的，用户的选择情况可以保存到变量中，最后生成代码时，根据用户的选择，对模板文件一次性修改完成，如图 5 是一个向导程序的界面。

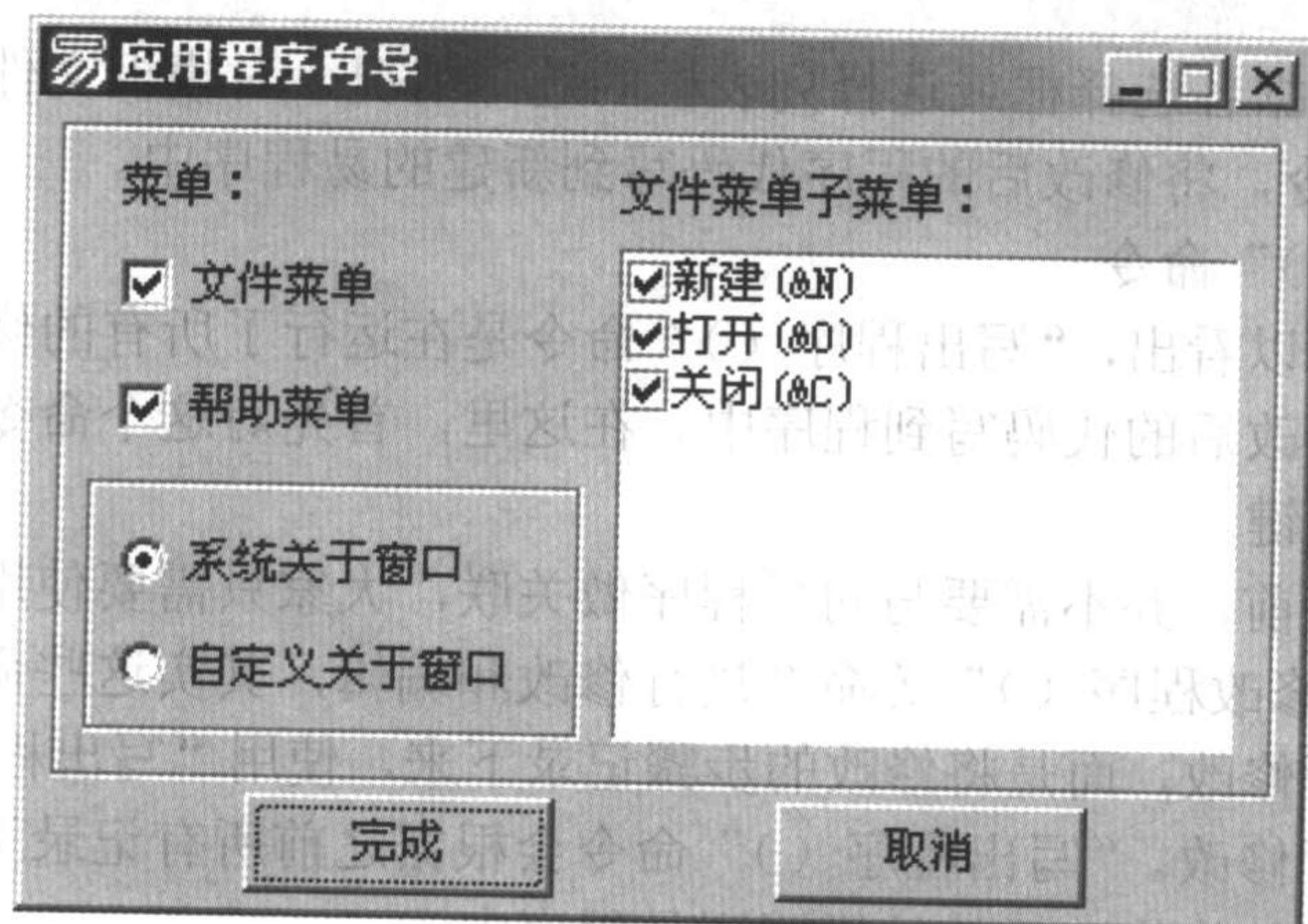


图 5 向导程序界面

该向导最终写出程序的代码如下：

子程序名	返回值类型	公开	备注
_完成按钮_被单击			开始修改模板

变量名	类型	静态	数组	备注
结果文本	文本型			

```

如果真 (文件菜单选择列表框.是否被选中 (0) = 假) ' 新建菜单是否被选中
    删除程序 (#程序项类型.窗口组件, "_启动窗口.新建")
如果真 (文件菜单选择列表框.是否被选中 (1) = 假) ' 打开菜单是否被选中
    删除程序 (#程序项类型.窗口组件, "_启动窗口.打开")
如果真 (文件菜单选择列表框.是否被选中 (2) = 假) ' 关闭菜单是否被选中
    删除程序 (#程序项类型.窗口组件, "_启动窗口.关闭")
    
```





```
如果真 (文件菜单选择列表框.是否被选中 (0) = 假 且 文件菜单选择列表框.是否被选中 (1) = 假 且 文件菜单选择列表框.是否被选中 (2) = 假 或 文件菜单选择框.选中 = 假)
    删除程序 (#程序项类型.窗口组件, “_启动窗口.文件”)
    如果真 (帮助菜单选择框.选中 = 假) ' 判断帮助菜单是否被选择
        删除程序 (#程序项类型.窗口组件, “_启动窗口.帮助”)
        删除程序 (#程序项类型.窗口组件, “_启动窗口.关于”)
    如果真 (自定义关于单选框.选中)
        删除程序 (#程序项类型.DLL命令, “ShellAbout”)
        删除程序段 (“_关于_被选择”, “dll_ShellAbout”)
    如果真 (系统关于单选框.选中)
        删除程序段 (“_关于_被选择”, “载入关于窗口”)
    结果文本 = 写出程序 (读入文件 (“c:\模板文件.e”),)
    ' 将修改后的源代码写到易程序中
    判断 (结果文本 = “”)
    信息框 (“已经创建 应用程序”, 0, )
    信息框 (结果文本, 0, )
    销毁 0
```

以上代码, 是根据各选择框或选择列表框的选择情况, 对模板文件进行修改, 最后用“写出程序 ()”命令, 将修改后的程序代码写到新建的易程序中。

#### 1. “写出程序 ()”命令

从上面的代码可以看出, “写出程序 ()”命令是在运行了所有的修改步骤后, 最后使用的命令, 用来将修改后的代码写到程序中。在这里, 首先对这个命令进行介绍, 因为该命令是向导程序的关键。

模板程序在写出程序前, 并不需要与向导程序做关联, 大家只需要使用“删除程序 ()”、“复制程序 ()”、“修改程序 ()”等命令进行修改和编辑, 其实这些命令使用时, 并没有直接对模板文件进行修改, 而是将修改的步骤记录下来, 使用“写出程序 ()”命令时, 才真正对模板文件进行修改。“写出程序 ()”命令会根据之前所有记录下来的修改步骤, 对模板文件进行修改, 并将修改后的结果写到易程序中。

要注意的是, 对模板文件的修改, 是将模板文件读到内存中进行的, 所以并不会修改原文件。

“写出程序 ()”命令有两个参数, 第 1 个参数为字节集型, 是读入的模板程序数据; 第 2 个参数为逻辑型, 代表是否关闭源程序, 如果为真, 向导程序会将当前打开的易程序关闭, 如果为假, 则不会关闭当前已打开的易程序。

注意, 只有第 2 个参数为假时, 向导程序才可以将代码插入到已打开的易程序中。插入的方法为使用菜单: 工具→执行易向导, 来执行指定的向导程序。

“写出程序 ()”命令返回在执行过程中出现错误的错误文本, 如果返回值为空文本, 则表示执行成功。

上面介绍的向导程序中, 写出程序的代码为:

```
结果文本 = 写出程序 (读入文件 (“c:\模板文件.e”),)
' 将修改后的源代码写到易程序中
```

上面代码使用的模板是存放在 C 盘根目录的, 直接读指定路径的模板文件有不足之处,



即如果当模板文件的位置改变时，向导程序就会出现错误，所以可以将模板文件存放在易语言“资源表”中，这样调用起来也很方便，并且如果不想公开模板文件的代码，也可以使用这种方法。例如，将上面代码中“C:\模板文件.e”存放在“资源表”中，则代码变为：

结果文本 = 写出程序 (#向导程序, )

将修改后的源代码写到易程序中

由于“资源表”中的数据本身就是字节集型的，所以直接调用即可。

## 2. “删除程序 ()”和“删除程序段 ()”命令

“删除程序 ()”命令，可以删除指定的程序项，如窗口组件、程序集、变量等，该命令有 2 个参数，第 1 个参数为程序项类型，需要填写“程序项类型”枚举常量中的成员；第 2 个参数为文本型，为欲删除项目的名称。

例 1，删除“\_启动窗口”中的“打开”菜单，代码如下：

删除程序 (#程序项类型.窗口组件, “\_启动窗口.打开”)

例 2，删除“窗口程序集 1”中的“\_\_启动窗口\_创建完毕”子程序，代码如下：

删除程序 (#程序项类型.子程序, “窗口程序集1.\_\_启动窗口\_创建完毕”)

“删除程序段 ()”命令，可以删除指定子程序中的一段代码，前提是此段代码必须做了标记。

注意：易向导程序所操作代码的标记是写在模板程序代码的备注中的。格式为“\$ (标记名称)”或“\$ (标记名称/)”加“\$ (/标记名称)”，第一个标记方法用来删除一行代码，后两种标记方法只能在程序语句中成对使用，用作定义具有多行语句的代码块标记。

“删除程序段 ()”命令有两个参数，第 1 个参数为删除代码所在的子程序名；第 2 个参数为标记名，注意：该参数只填写标记中括号之间的标记名部分。

例 1，删除模板程序的“\_关于\_被选择”子程序中标记为“\$ (载入关于窗口)”的代码。模板程序中的代码如下：

子程序名	返回值类型	公开	备注
_关于_被选择			

载入 (关于窗口, , 真) ' \$ (载入关于窗口)

ShellAbout (0, “XXX”, “由 XXX 制作”, 0) ' \$ (dll\_ShellAbout)

删除标记名为“载入关于窗口”的代码行，向导程序代码如下：

删除程序段 (“\_关于\_被选择”, “载入关于窗口”)

例 2，删除模板程序中，“\_关于\_被选择”子程序，标记名为“自定义关于窗口”的程序块，模板代码如下：

子程序名	返回值类型	公开	备注
_关于_被选择			

载入 (关于窗口, , 真) ' \$ (自定义关于窗口/)

关于窗口.标签1.标题 = “XX 作者版权所有”

关于窗口.编辑框1.内容 = #软件条款

关于窗口.按钮1.标题 = “关闭” ' \$ (/自定义关于窗口)

ShellAbout (0, “XXX”, “由 XXX 制作”, 0) ' \$ (dll\_ShellAbout)

删除标记名为“自定义关于窗口”代码块的向导程序代码如下：





## 删除程序段 (“\_关于\_被选择”, “自定义关于窗口”)

易向导支持库中, 其他涉及标记定义和使用的命令, 定义方法和使用方法都和“删除程序段 ()”命令相同, 大家可以根据该命令中的介绍, 来使用其他的一些命令。

### 3. “修改程序 ()”命令

该命令用来修改模板程序中的指定程序项的名称或属性等。

该命令有 4 个参数, 第 1 个参数为程序项类型, 需要填写“程序项类型”枚举常量中的成员; 第 2 个参数为欲修改的程序项名称; 第 3 个参数为程序项属性, 需要填写“程序项属性”枚举常量中的成员; 第 4 个参数为欲修改的值, 通用型, 该参数应该填写与被修改的项目相同数据类型的数据。

例 1, 将模板中, “窗口程序集 1” 的名称该为 “\_启动窗口程序集”, 代码如下

```
修改程序 (#程序项类型. 程序集, “窗口程序集1”, #程序项属性. 名称, “_启动窗口程序集”)
```

例 2, 将模板中, “类 1” 的 “比较大小” 方法的公开属性设置为真, 代码如下:

```
修改程序 (#程序项类型. 子程序, “类1. 比较大小”, #程序项属性. 公开, 真)
```

### 4. “置组件属性 ()”命令

该命令可以用来修改模板程序中, 组件的属性。

该命令有两个参数, 第 1 个参数为组件的名称, 文本型, 组件属性名以 “窗口名. 组件名. 属性名” 或 “窗口名. 属性名” 格式表示, 注意窗口组件包括菜单项; 第 2 个参数为欲修改的属性值, 通用型, 该参数应该填写与被修改属性相同数据类型的数据。

例 1, 将模板程序中 “窗口 1” 的 “编辑框 1” 的 “可视” 属性设置为 “假”, 代码如下:

```
置组件属性 (“窗口1. 编辑框1. 可视”, 假)
```

例 2, 将模板程序中 “窗口 1” 的 “标题” 属性设置为 “易程序”, 代码如下:

```
置组件属性 (“窗口1. 标题”, “易程序”)
```

### 5. “清除修改记录 ()”命令

上面提到过, 在使用 “写出程序 ()” 命令前, 进行的所有的删除和修改, 都只是做了修改记录, 并没有在模板程序上直接进行修改, 当 “写出程序 ()” 命令运行时才按照上面所有的修改记录, 对模板进行修改。

“清除修改记录 ()” 命令, 先前所有修改记录都清除掉, 该命令运行后, 等于未对模板程序进行任何修改。

### 6. “定义模板变量 ()”命令

定义模板变量, 是在向导程序中, 为模板程序定义一个变量, 这个变量只在生成代码的瞬间起作用, 生成后的代码中是看不到的。

在模板程序中可以使用 “\$如果 (条件)”、“\$否则 (条件)” 这样的条件语句 (在备注中使用) 对模板变量进行判断, 在条件语句的条件中加入对模板变量的判断, 便可以进行大范围代码的删除和修改。

当 “\$如果 ()” 语句条件中的值为真, 则条件成立, 并保留 “\$如果 ()” 语句中的代码, 如果条件中的值为假, 则条件不成立, 将删除 “\$如果 ()” 语句中的代码; “\$否则 ()” 语句必须使用在 “\$如果 ()” 语句之后, 用来判断 “\$如果 ()” 条件不成立时的其他条件, 使用方法和 “\$如果 ()” 语句基本相同。

注意, “\$如果 ()” 和 “\$否则” 等, 带有 “\$” 号的语句, 都是必须在备注中使用的语



句，后面不再做特殊声明。

使用模板变量会给制作向导带来事半功倍的效果，下面就来详细介绍如何使用模板变量。

“定义模板变量 ( )”命令的作用很简单，就是在模板程序中创建一个模板变量。命令有两个参数，第 1 个参数为模板变量的名称，文本型；第 2 个参数为模板变量的值，通用型，当模板变量定义后，模板变量的数据类型跟赋予的值的类型相同。

模板变量的使用，主要并不是在向导程序中，向导程序中只负责对模板变量进行定义，而对模板变量的判断，是在模板程序的代码中，所以，模板程序中要对并不存在的模板变量进行判断，只有当代码写出前，该模板变量才会被定义并起作用。下面将举例说明模板变量的使用方法。

(1) 对模板变量是否存在进行判断，从而删除指定段代码。

例如，在模板程序中的代码如下：

子程序名	返回值类型	公开	备注
_关于_被选择			\$如果 (!删除关于菜单)

' \$如果 (自定关于窗口)

载入 (关于窗口, , 真)

' \$如果 (系统关于窗口)

ShellAbout (0, "XXX", "由 XXX 制作", 0)

当“\$如果 ( )”语句中，不对模板变量的值进行判断，而只将模板变量的名字写在条件中时，则“\$如果 ( )”语句会对模板变量是否存在进行判断，如果条件中的模板变量存在，则返回真，条件成立；如果条件中的模板变量不存在，则返回假，条件不成立。

首先看子程序中代码中“\$如果 ( )”语句的条件，该条语句对“自定义关于窗口”和“系统关于窗口”这两个模板变量是否存在进行了判断，当条件中的模板变量被定义，则会保留“\$如果 ( )”中的代码。

根据向导窗口中，单选框的选择情况，来保留指定的关于窗口的代码如下：

```

-- 如果真 (自定义关于单选框.选中 = 真)
    定义模板变量 ("自定义关于窗口", )
-- 如果真 (系统关于单选框.选中 = 真)
    定义模板变量 ("系统关于窗口", )

```

接下来，大家看上面模板程序中，“\_关于\_被选择”子程序的备注：“\$如果 (!删除关于菜单)”该语句的条件中，在模板变量的名称前面加了“!”号，表示取相反的结果，则该条件就表示，当“删除关于菜单”模板变量不存在时，保留该子程序，如果定义了该模板变量，则删除该子程序。

(2) 对模板变量的值进行判断。模板变量可以被赋予指定的值，并且模板变量的数据类型和被赋值的类型相同，即给模板变量赋一个文本值，则模板变量就为文本型。所以在对模板变量值进行判断时，要注意模板变量值的数据类型。

例如，在向导程序中，提供一个组合框，让用户选择需要哪种风格的 XP 风格界面，从而根据用户在向导中的选择，来保留模板指定的 XP 风格界面的代码。组合框提供的选项如图 6 所示。



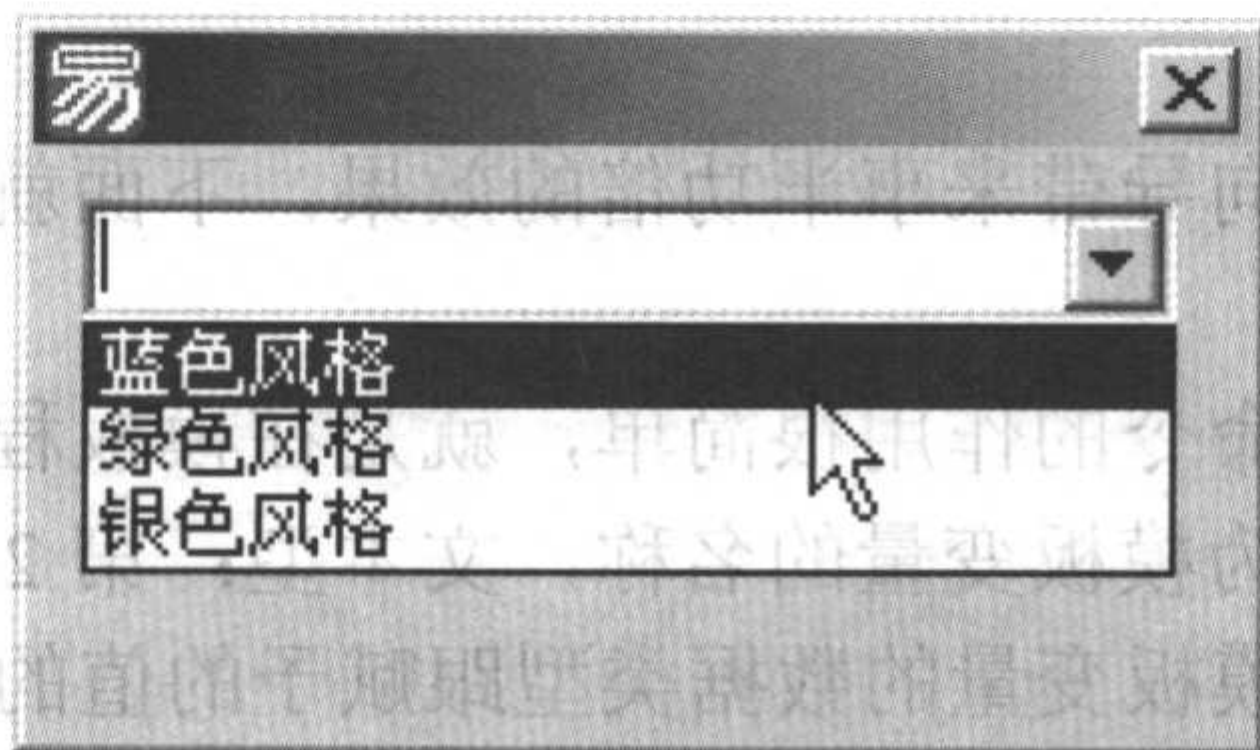


图6 选择XP风格的组合框

在向导程序中，定义一个模板变量“XP 风格”，将用户选择的项目文本赋值给该模板变量，向导程序中代码如下：

定义模板变量（“XP风格”，XP风格组合框.取项目文本（XP风格组合框.现行选中项））

在定义模板变量时，将组合框的项目文本赋值给模板变量，所以，该模板变量为文本型，则在模板程序中对模板变量值进行判断时就应该注意文本型数据的表示方法。

在模板程序中，对“XP 风格”模板变量的判断代码如下：

```
' $如果 (XP风格=“蓝色风格”)
XP风格 (#蓝色风格)
' $否则 (XP风格=“绿色风格”)
XP风格 (#绿色风格)
' $否则 (XP风格=“银色风格”)
XP风格 (#银色风格)
' $结束
```

程序代码中使用了“\$如果 ()”、“\$否则 ()”和“\$结束”语句，这三条语句是配合使用的，其中“\$否则 ()”语句必须在“\$如果 ()”语句之后使用，用来判断当“\$如果 ()”中条件不成立时的其他条件，当“\$如果 ()”中的条件成立，则“\$否则 ()”语句不会进行判断，其中的代码将会被删除，在进行完一组判断后，一定要使用“\$结束”语句，用来表示一组条件语句的结束，“\$结束”语句使用后，下面的代码可以使用另一组判断，而不受前面条件语句的影响。

上面的例程也可以改为对整数值进行判断，可以将组合框的现行选中项的索引赋值给模板变量，则模板变量会变为整数型。注意：在对整数型模板变量进行判断时，直接用“模板变量=XXX”的形式表示，不需要加双引号。

(3) 直接引用模板变量的值，动态的改变模板程序中的代码。

当模板变量被赋值后，则该值在写出代码的时，可以被引用，可以用该值来改变模板程序代码中的属性值、变量值等等。引用方法为：“\$（模板变量名）”（双引号保留）。

例1，以上选择XP风格例程中的一组判断可以使用一行代码来代替，向导程序中定义模板变量的方法如下：

定义模板变量（“XP风格”，XP风格组合框.现行选中项 + 1）

由于在模板程序中，将引用该模板变量的值，组合框的现行选中项的项目索引从0开始，而引用的值需要从1开始，所以给模板变量赋值时将项目索引加1。

模板代码中的代码如下：



### XP风格 (“\$XP风格”)

代码中，直接使用“XP 风格”模板变量的值填写“XP 风格( )”命令的参数。注意，这样编写代码，一定要保证定义了模板变量，否则会将当前代码的原样写到程序中。

例 2，易语言自带的应用程序向导，用户可以选择“资源管理器样式”的界面，生成后的程序，不管用户是否选择了保留工具条或状态条，组件的大小都可以跟随窗口大小的改变而改变。该向导程序中的一段代码如下：

```

--- 判断 (资源管理器样式单选框.选中 = 真)
    --- 判断 (工具条选择框.选中 = 假 且 状态条选择框.选中 = 假)
        --- 定义模板变量 (#资源管理器, 0)
        --- 判断 (工具条选择框.选中 = 假 且 状态条选择框.选中 = 真)
            --- 定义模板变量 (#资源管理器, 25)
            --- 判断 (状态条选择框.选中 = 假 且 工具条选择框.选中 = 真)
                --- 定义模板变量 (#资源管理器, 53)
                --- 定义模板变量 (#资源管理器, 53 + 20)
    
```

向导程序中，对用户是否选择保留状态条和工具条进行判断，从而为“资源管理器”模板变量赋不同的值。

在模板程序中，“\_\_启动窗口\_尺寸被改变”子程序代码如下：

子程序名	返回值类型	公开	备注
__启动窗口_尺寸被改变			\$如果(资源管理器)
树型框1.高度 = 取用户区高度 0 - 19 - “\$(资源管理器)”			
超级列表框1.移动 (, 取用户区宽度 0 - 分隔条1.左边 - 分隔条1.宽度, 取用户区高度 0 - 19 - “\$(资源管理器)” )			
超级列表框1.高度 = 取用户区高度 0 - 19 - “\$(资源管理器)”			
分隔条1.高度 = 取用户区高度 0 - 19 - “\$(资源管理器)”			

当\_启动窗口的尺寸被改变，则窗口中组件的大小随之改变，由于不知道用户最终是否选择保留状态条和工具条，所以组件移动时都将“高度”属性减“资源管理器”模板变量的值，由于用户选择的不同，“资源管理器”模板变量的值不同，便实现了用户选择的不同，生成的改变窗口组件大小的代码也不同。

#### (4) 模板变量在不同备注中的判断。

模板变量在代码中，单独在一行中的备注中使用“\$如果( )”等语句进行判断。而“\$如果( )”语句不但可以使用在代码的备注中，并且可以使用在任何程序项的备注中，例如，常量表、DLL 命令声明、组件、资源表中的资源，都可以在其备注中使用“\$如果( )”语句进行判断，来确定是否保留该程序项。

例如，根据用户是否选择保留“登录窗口”，来删除和保留模板程序中的登录窗口，在登录窗口的备注中，便可以对指定的模板变量进行判断，在向导程序中，可以定义指定的模板变量来删除和保留登录窗口。如图 7 所示。



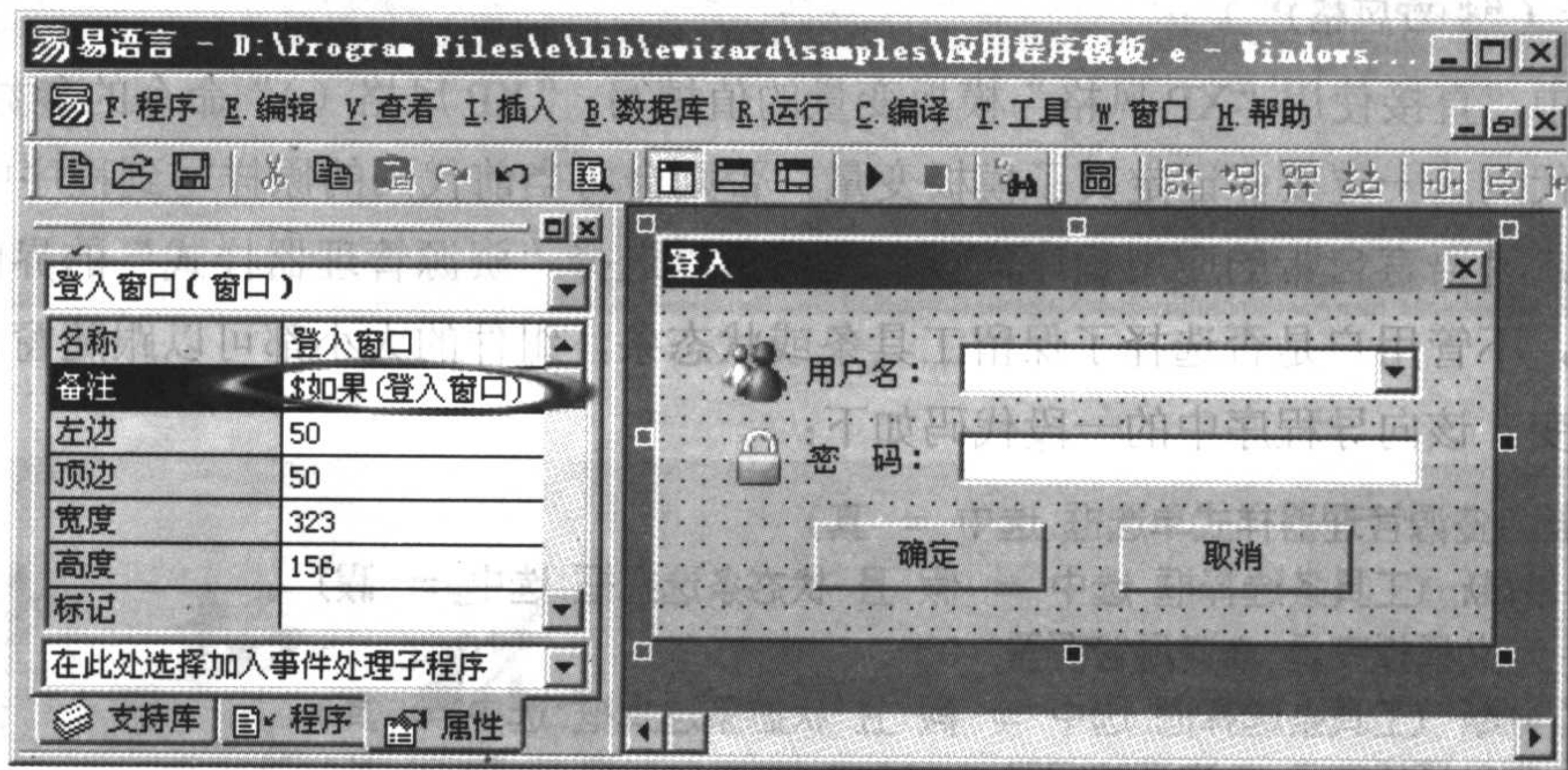


图 7 在窗口备注中进行判断

在模板程序中，所有有备注的程序项都可以使用“\$如果 ()”语句对模板变量进行判断。

易向导支持库的使用方法就介绍到这，本支持库可以很方便地实现对源代码的编辑，有助于易语言爱好者互相交流编程经验，希望大家多编写向导程序，方便自己 and 他人。

在编写易向导时，还需要注意以下几个方面：

1. 在向当前打开易程序中插入代码时，插入代码中的程序项目和当前打开易程序中程序项重名时，插入的代码的程序项会自动更名，所以要注意使用“\_启动窗口”等常用的名称。例如模板程序中有“\_启动窗口”，当前打开的易程序中也有“\_启动窗口”，则插入到当前打开易程序中的“\_启动窗口”会自动更名。
2. 易向导支持库中很多命令都有“程序项名称属”，这些参数都要使用“父项目.子项目”的形式表示。例如，“删除程序 ()”、“复制程序 ()”命令。
3. 如果要制作可以在当前打开易程序中插入代码的易向导，要将“写出程序 ()”命令的第二个参数设置为“假”，并且要注意程序项名称不要与“\_启动窗口”等经常使用的程序项重名。
4. 模板程序代码中的条件语句最好写在空行中的备注中。



[ G e n e r a l   I n f o r m a t i o n ]

书名 = 易语言编程系统

作者 = 易语言教材编委会编著

页数 = 5 2 4

S S 号 = 1 1 6 4 8 8 7 7

出版日期 = 2 0 0 5 | 出版社 = 西安地图出版社



封面  
书名  
版权  
前言  
目录  
正文